Detecting Polycystic Ovary Syndrome (PCOS) Using ML Techniques

Machine Learning Paper Implementation Report

Submitted by:
E061: Khushi Vora
E071: Arya Shah

Under the Guidance of:
Prof. Priyanka Verma

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN COMPUTER SCIENCE & BUSINESS SYSTEMS

At
NMIMS UNIVERSITY, MUMBAI

OCTOBER, 2021

Declaration

We, **Khushi Vora, Arya Shah, Roll Nos. E061, E071 of** B.Tech (Computer Science and Business Systems), V semester understand that plagiarism is defined as anyone or combination of the following:

1. Un-credited verbatim copying of individual sentences, paragraphs or illustration (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.

2. Un-credited improper paraphrasing of pages, paragraphs (changing a few words phrases, or rearranging the original sentence order)

3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did write what. (Source: IEEE, The institute, Dec. 2004)

4. I have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.

5. I affirm that no portion of my work can be considered as plagiarism and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the seminar/ project report may not be in a position to check for the possibility of such incidents of plagiarism in this body of work.

Certificate

This is to certify that the project entitled "**Detecting Polycystic Ovary Syndrome (PCOS) Using ML Techniques**" is the bonafide work carried out by **Khushi Vora, Arya Shah** of B.Tech (Computer Science and Business Systems), MPSTME (NMIMS), Mumbai, during the V semester of the academic year 2020-21, in partial fulfillment of the requirements for the award of the Degree of Bachelors of Technology as per the norms prescribed by NMIMS. The project work has been assessed and found to be satisfactory.

____

Prof. Priyanka Verma

ABSTRACT

The present world women population is widely affected by preterm abortions, infertility, anovulation etc. It is observed that polycystic ovary syndrome (PCOS), a condition seen among the women of reproductive age, is having a major influence in the cause of infertility. Over five million women worldwide in their reproductive age PCOS. It is an endocrine disorder characterized by changes in the female hormone levels and the abnormal production of male hormones. This condition leads to ovarian dysfunction with increased risk of miscarriage and infertility. The symptoms of PCOS include obesity, irregular menstrual cycle, and excessive production of male hormone, acne, and hirsutism. It is extremely difficult to diagnose PCOS due to the heterogeneity of symptoms associated and the presence of a varying number of associated gynecological disorders. The time and cost involved in innumerous clinical tests and ovary scanning has become a burden to the patients with PCOS. To address this problem this paper proposes a system for the early detection and prediction of PCOS from optimal and minimal but promising clinical and metabolic parameters, which act as an early marker for this disease. The data sets required for this system development are obtained through patient surveys of 541 women during doctor consultations and clinical examinations. Out of the 23 features from clinical and metabolic test results, 8 potential features are identified using SPSS V 22.0 based on their significance. Classification of PCOS with the feature set transformed with Principal Component Analysis (PCA) is done using various machine learning techniques such as Naïve Bayes classifier method, logistic regression, K-Nearest neighbor (KNN), Random Forest Classifier, Support Vector Machine (SVM) in Python. Results revealed that the most suitable and accurate method for the PCOS prediction is RFC with an accuracy of 89.02%.

Keywords—Machine learning, polycystic ovary syndrome, Classifier, Diagnostic aid

i-HOPE: Detection And Prediction System For Polycystic Ovary Syndrome (PCOS) Using Machine Learning Techniques

Amsy Denny
Dept. of Biomedical Engineering, Sahrdaya College of Engineering and Technology (Affiliated to APJ Abdul Kalam Technological University) Thrissur, India
amsydennykallingal@gmail.com

Anita Raj
Dept. of Biomedical Engineering, Sahrdaya College of Engineering and Technology (Affiliated to APJ Abdul Kalam Technological University) Thrissur, India
anitakallery@gmail.com

Ashi Ashok
Dept. of Biomedical Engineering, Sahrdaya College of Engineering and Technology (Affiliated to APJ Abdul Kalam Technological University) Thrissur, India
ashisgashok@gmail.com

Maneesh Ram C
Dept. of Biomedical Engineering, Sahrdaya College of Engineering and Technology (Affiliated to APJ Abdul Kalam Technological University) Thrissur, India
maneeshram99@gmail.com

Remya George
Dept. of Biomedical Engineering, Sahrdaya College of Engineering and Technology (Affiliated to APJ Abdul Kalam Technological University) Thrissur, India
remyageorge@sahrdaya.ac.in

Abstract—The present world women population is widely affected by preterm abortions, infertility, anovulation etc. It is observed that polycystic ovary syndrome (PCOS), a condition seen among the women of reproductive age is having a major influence in the cause of infertility. Over five million women worldwide in their reproductive age PCOS. It is an endocrine disorder characterized by changes in the female hormone levels and the abnormal production of male hormones. This condition leads to ovarian dysfunction with increased risk of miscarriage and infertility. The symptoms of PCOS include obesity, irregular menstrual cycle, and excessive production of male hormone, acne, and hirsutism. It is extremely difficult to diagnose PCOS due to the heterogeneity of symptoms associated and the presence of a varying number of associated gynecological disorders. The time and cost involved in innumerous clinical tests and ovary scanning has become a burden to the patients with PCOS. To address this problem this paper proposes system for the early detection and prediction of PCOS from an optimal and minimal but promising clinical and metabolic parameters, which act as an early marker for this disease. The data sets required for this system development are obtained through patient survey of 541 women during doctor consultations and clinical examinations. Out of the 23 features from clinical and metabolic test results, 8 potential features are identified using SPSS V 22.0 based on their significance. Classification of PCOS with the feature set transformed with Principal Component Analysis (PCA) is done using various machine learning techniques such as Naïve Bayes classifier method, logistic regression, K-Nearest neighbor (KNN), Classification and Regression Trees (CART), Random Forest Classifier, Support Vector Machine (SVM) in Spyder Python IDE. Results revealed that the most suitable and accurate method for the PCOS prediction is RFC with an accuracy of 89.02%.

Keywords—Machine learning, polycystic ovary syndrome, Classifier, Diagnostic aid.

I. Introduction

Technology and mankind together hand in hand can make way towards better health care and services. Machine learning is a subset of artificial intelligence, in which it provides the system with the ability to automatically learn and improve without being programmed explicitly. It mainly focuses on developing algorithms that can access the datasets provided and use data for the learning purposes of the network. Applications of Machine Learning bring about huge transformation in the health industry, which includes detection, data prediction, image recognition etc.

Polycystic ovary syndrome (PCOS), is one of the relevant, most prevalent hormonal disorder seen among the women of childbearing age. This is a heterogeneous endocrine disorder which is highly prone to infertility, anovulation, cardiovascular disease, type 2 diabetes, obesity etc. PCOS is a common condition detected in nearly 12-21% of women of reproductive age and among them 70% is remain undiagnosed. PCOS condition can be treated to some extend by controlled medication and bringing alterations in life style. This includes the treatment methods with pills for birth control, diabetes, fertility, anti-androgen medicines and scanning procedures like ultrasound scan. When such interventions fail, invasive treatment procedures like surgical drilling of ovaries is also used for improving the ovulation ability of the ovary by reducing the male hormone level.

The aetiology of PCOS is underpinned by both insulin resistance and hyperandrogenism. Clinically it is characterized by reproductive, metabolic and psychological features and represents a major health burden to women. Diagnosis is recommended based on clinical or biochemical and radiological test results. PCOS is diagnosed by exclusion of irrelevant symptoms or test results, mainly because of lack of knowledge of its complex patho-mechanism. The diverse symptoms of this condition force medical practitioners to call for large number of clinical test results and unnecessary radiological imaging procedures. The early detection and diagnosis of PCOS with minimal tests and imaging procedures is of utmost importance and of great significance as the condition directly leads to ovarian dysfunction with an increased risk of miscarriage, infertility or even gynaecological cancer and mental agony for the patients due to wastage of time and money.

II. Literature Survey

Among the in-numerous problems that exist around us, the problems that are related to the reproductive health of women

978-1-7281-1895-6/19/$31.00 ©2019 IEEE

was selected as an area of our interest, due to its importance in this contemporary society. A detailed survey of studies on PCOS and systems to support its diagnosis was carried out. Literature says that about 5-10% of Indian women in reproductive age are affected by the multifaceted endocrine disorder called Polycystic Ovary Syndrome (PCOS) [16]. It is a major cause of anovulatory infertility and increases the risk for insulin resistance, obesity, cardiovascular disease and psychosocial disorders [17]. The symptoms for PCOS might be varying from patient to patient. Some of them are irregularity in menstrual periods, acne, overweight, increased tendency for infertility, intense hair fall, balding of front head, increased facial hair growth [1]. Traditionally the PCOS can be suspected when number of follicles in an ovary is more than 12 per unit area and visible in radiological scan [15]. Some authors have proposed changing the cutoff from 12 follicles to 20 or abandoning ultrasound altogether in favor of other biomarkers, such as serum anti-Mullerian hormone (AMH) [1,10]. The diagnosis of PCOS is uncomplicated, requiring only the careful application of a few well-standardized diagnostic methods. PCOS diagnosis is often delayed and this affects patients' well-being negatively [21]. Escobar *et al.* [11] suggests that treatment should be symptom-oriented, long term and dynamic and adapted to the changing circumstances, personal needs and expectations of the individual patient. Joham et al. in [6] considered the relation of PCOS and infertility rate of women in this community and use of fertility hormone treatment was significantly higher in women reporting PCOS. Considering the prevalence of PCOS and the health and economic burden of infertility, strategies to optimize diagnosis of PCOS and the factors leading to fertility are important. This is because iinfertility is reported to be 15-fold higher in women reporting PCOS, independent of BMI [18]. There is a bi‑directional relationship between obesity and PCOS. Both exacerbate each other in a never‑ending cyclical manner. Essah, P.A. and Nestler, J.E suggests that the prevalence of obesity in PCOS women is 30–75% [19]. Clinical validation of PCOS is usually done by Rotterdam criteria [8] or standards set by societies involved in PCOS research. Pictorial depiction of three popular criteria can be seen in Table I.

TABLE I. CRITERIA FOR DIAGNOSIS OF PCOS

Clinical Finding	National Institutes of Health criteria,1990 (Must have both of the findings marked below)	Rotterdam Criteria, 2003(must have any two of the findings marked below)	Androgen Excess and PCOS Society, 2009(must have A plus either B or C)
*Hyperandrogenism**	X	X	A
Oligomenorrhea	X	X	B
Polycystic ovaries		X	C

* Clinical or biochemical evidence of excess androgen.

A cross sectional study by Brower *et al.* [14] suggested that the presence of clinically evident menstrual dysfunction can be used to predict the presence and possibly the degree of insulin resistance in women with PCOS. Many of the technical studies carried out in PCOS diagnosis are using features of the ultrasound scan and image processing techniques for the diagnosis of the PCOS [1-2]. Some studies used clinical and metabolic features of the disease [3]. Few recent studies are diverted in fundamental research direction, investigating the associated factors such as obesity [4] and genetic factors [5]. A summary of such studies are given in Table II. Studies are also carried out in directions of analyzing urinary steroid hormone metabolites and enzyme activities in women with and without PCOS in order to test their value for diagnosing PCOS [20].

TABLE II. SUMMARY OF FEW STUDIES ADDESSING THE ISSUE OF PCOS

Authors	Technique used	Objective of the study	Year
Cheng et al. [1]	Rule based classifier and Gradient boosted Tree classifiers	PCOS determination from Ultrasound Images	2016
Dewi, R.M. and Wisesty [2]	Gabor wavelet based feature extraction and CNN	PCOS determination from Ultrasound Images.	2018
Mehrotra et al. [3]	2 sample-test, Bayesian and Logistic Regression (LR) classifier	PCOS determination from clinical and metabolic parameters.	2011
Sachdev et al. [4]	Prospective observational study : Obese vs non-obese PCOS	Obese PCOS patients have a higher risk of adverse outcomes.	2019
Zhang et al. [5]	Machine-learning algorithms	PCOS prediction from identification of new PCOS genes.	2019
Joham et al. [6]	Logistic regression on data from cross-sectional analysis of a longitudinal cohort	Examination of factors associated with infertility and use of fertility treatment.	2015
K. Meena, M. Manimekalai, and S. Rethinavalli [7]	Information Gain Subset Evaluation and Neuro Fuzzy methods of feature selection and Decision Tree classifier	Framework for Filtering the PCOS Attributes	2015

III. METHODOLOGY

For the development of an appropriate machine learning model based diagnostic aid for PCOS, a comparison of performance of various existing algorithms in our data set need to be presented. Preparation of the model is the most crucial step that provides the outline of the research. Steps that are included in the development of an appropriate model and tuning it for obtaining possibly the best result, is detailed below with the help of a work flow diagram, Figure 1. Along with that the effective tools and available platforms utilized for the development of the system must be mentioned. The following section describe both aspects.

A. Defining problem

The most important step is to define the problem appropriately including the inputs provided into the model and the output expected out of it. It is based on the assumptions like the outputs can be predicted from the inputs provided.

B. Data collection

The critical step that will decide about how good the model will be and also as the number of data collected increases the accuracy of the model also increases and hence better will be its performance. There are many ways for data gathering like real time data gathering or from repository platforms like kaggle and UCI machine learning repository which is one of the most frequently visited one.

C. Selection of implementation platform

Tools for the efficient running of machine learning methods and also platforms for statistical analysis should be properly opted. For this research, Sypder python for model formation, HTML with SQL for designing a proper user interface, whereby the patient data can be input to the system and PCOS status can be obtained as output, and SPSS V22.0 for establishing the relevance of features, are used.

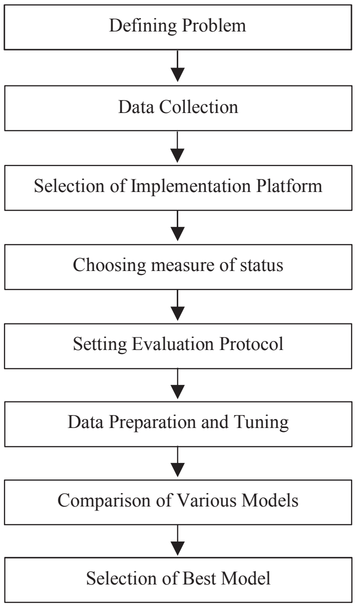

Defining Problem
Data Collection
Selection of Implementation Platform
Choosing measure of status
Setting Evaluation Protocol
Data Preparation and Tuning
Comparison of Various Models
Selection of Best Model

Fig. 1. Workflow of machine learning model

D. Choosing measure of status

In case classification problems, success is measured using the calculation of accuracy and precision of the model. In this study we have considered the following evaluation metrics:

- Accuracy : $\frac{(\mathrm{TP+TN})}{(\mathrm{TP+TN+FP+FN})}$
- Precision (P) : $\frac{(\mathrm{TP})}{(\mathrm{TP+FP})}$
- Sensitivity / Recall (R) : $\frac{(\mathrm{TP})}{(\mathrm{TP+FN})}$
- Specificity : $\frac{(\mathrm{TN})}{(TN+FP)}$
- F1 score : $2 * \left(\frac{P*R}{P+R}\right)$

where TP, TN, FP, FN implies True Positive, True Negative, False Positive, False Negative respectively.

E. Setting validation protocol

Maintaining a hold out validation set, i.e., in this method some portion of the data is set apart for the purpose of testing as test data and remaining as train data.

Usually the data is split in the ratio 8:2 as train data to test data. It can be depicted as in fig.2:

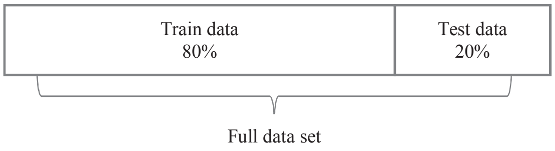


Fig. 2. Depiction of Hold-out validation set

F. Data preparation

This tiring process includes dealing with missing data, handling categorical data, feature scaling and selection of meaningful features. The missing values in the dataset is replaced by 'NaN'. Due to a model's inability to read a missing value, before confronting the model, the samples with missing values will be extirpated or else will be replaced with some pre-built estimators. Likewise before feeding the data into the model, ordinal and nominal data need to be considered accordingly. If the nature of the data demands, the dataset should undergo normalization and standardization. Finally, the overfitting can be avoided by reducing the dimensionality of data. This is done by reducing the number of feature sets present in the dataset. It is performed using Principal Component Analysis (PCA) in Spyder Python IDE, which works by identifying the patterns in the datasets and the correlations present between the features. The correlated data are then eliminated by directly removing such features. The optimal features identified by the PCA algorithm are verified for their potential in discriminating PCOS status, with SPSS V22.0. This is done by a procedure called independent sample t-test and the level of statistical significance across the two classes of PCOS and Non PCOS patients. Those features with a significance less than 0.01 are potential ones.

G. Comparison of various models

This step is to serve as a baseline. Study is carried out with selected set of features in a number of classifier algorithms. Among the existing innumerable machine learning algorithms some of them, which are proven to give best result in the detection of PCOS and Non-PCOS condition from the literature survey, is used and listed below:

- Logistic Regression (LR)
- Linear Discriminant Analysis (LDA)
- K-nearest neighbors (KNN)
- Classification and Regression Trees (CART)
- Random Forest Classifier
- Naïve Bayes Classifier
- Support Vector Machine

IV. DESIGN AND DEVELOPMENT

Data acquired from various hospitals and clinics which includes both physiological and metabolic parameters that are the contributors towards PCOS and thereby infertility, are fed into the mathematical framework of i-HOPE. The proposed system of diagnostic aid is illustrated Figure 3.

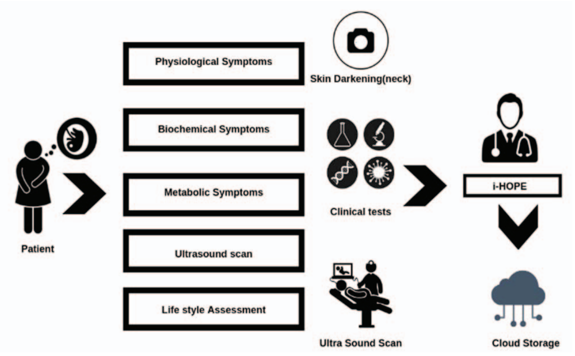


Fig. 3. Proposed model of i-HOPE

A. Data Collection :

More the number of features and samples, more distinguishing and accurate the model be. In this research we have considered a total of 541 samples which were collected from various clinics and hospitals in and around the district of Thrissur. Informed consent was collected from each patient, promising the anonymity of the data collected.

1) Patient History Collection :

One proforma is assigned for each patient so that all the data about a single patient will be in contained in a Proforma. These Proforma were carried along for data collection into clinics and for patient survey. Later on the collected data from the Proforma is consolidated together in to a single data sheet. The features which were included within the proforma was inferred from various literature studies and clinical surveys conducted, whose identity was kept anonymous during the entire process.

2) Real-time data collection :

For the purpose of real time detection of PCOS, a patient interface is required to input the patient data. Data to be entered may include the personal data of the patient for further references and the parameters that determine the existence of PCOS. This entered data will be inputting in to the created algorithm for data processing. To create the interface, a front end is generated using HTML and the data entered in the front end is being written in to an excel sheet using SQL.

TABLE III. FINALIZED PARAMETERS

SI No	Parameters	Value
1	Age	15-35
2	BMI	<24(normal), >24(abnormal)
3	Cycle Length and regularity	Long, normal or short Regular/Irregular
4	LH : FSH Ratio	Normal/abnormal
5	Waist : Hip Ratio	Normal/abnormal
6	Weight gain	Yes(y)/No(n)
7	Excess facial or body hair	Yes(y)/No(n)
8	Dark areas on skin	Yes(y)/No(n)
9	Pimples	Yes(y)/No(n)
10	Blood Pressure	Normal/abnormal
11	Diabetes (before and after food)	Normal/abnormal
12	Fast food intake	Yes(y)/No(n)
13	Regular exercise	Yes(y)/No(n)
14	Loss of hair	Yes(y)/No(n)
15	No. of follicles (L[a.] and R[b.])	High ,medium, low
16	Size of follicles (L[a.] and R[b.]) (mm)	>10
17	TSH (mIU/L)	0.4-4(Normal)

18	AMH	1-4(Normal)
		>1(Abnormal)

19	PRL	2-29 (Non-Pregnant Females) 10-209 (Pregnant Females)
20	Vit D3	20-50 (Normal) >12 (Abnormal)
21	PRG	1.5-12.4 (Normal)

[a.] Left ovary, [b.] Right ovary

B. Parameter Selection

The proposed model i-Hope was fed with the data obtained from the survey conducted. Parameter finalization was done with the support of expert opinions and considering the contemporary researches that in a way or other affected PCOS. The features are transformed for removing correlating among themselves which might adversely affect the classification results The finalized list of parameters before transformation are listed in Table III. These parameters include physiological, metabolic and biochemical attributes. Also considered the values of the result obtained from ultra sound scan which consists of the information about the cyst formed such as the number of cyst present in the right and left ovary and their size. The significance of parameters were studied individually based on the independent sample test, Pearson and Spearman's rho correlation of parameters with the help of SPSS software by IBM. The correlation was proved to be significant at 0.01 level (2-tailed). Table IV shows the important features for the final design of the system and their statistical significance.

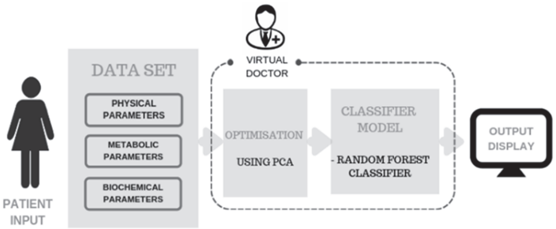


Fig. 4 The automated PCOS diagnosis aid i-Hope

C. Moulding Model

1) Setting model : Through Sypder python the evalution metrics of the model is claculated.After that the evaluation protocol is implemented from the sklearn library of the python platform. The most discriminating and contributing feature set is selected by extirpating the redundant data set through implementation of Principal Component Analysis (PCA) which is a type of unsupervised learning.

b.

2) Model Selection : Comparing various selected machine learning algorithms to find the best performing model using spot-check algorithm, which is a precursor for the selection of best and appropriate algorithm that suites the model. Selecting the best model with better performance by analyzing the confusion matrix of each algorithm. Among the models RFC provided the best result. Consolidating all the above noted details, the final design is improvised and modeled as shown in the Figure 4.

V. RESULT AND DISCUSSIONS

A total of 541 cases were available for study, which was collected from various infertility treatment centers at Thrissur. The data comprised the women reproductive age group i.e., in between 18-40 years. Among the data collected 364 cases were normal and non-PCOS, the remaining 177 cases reported PCOS. Altogether there were 23 features, including the reports on transvaginal Ultrasound scan, hormone profile and lifestyle of the patient with impressions on physical fitness and some listed in Table III. The optimal features after PCA is statistically analyzed to see their significance and are listed in Table IV.

The algorithms opted comprises a mixture of simple linear and non-linear algorithms. The simple linear algorithms are LR and LDA. Non-linear methods are KNN, CART, RFC, NB, SVM. We reset the random number seed in each run to the data split. Accuracy estimations of each of the models were carried out with hold out validation and the estimated accuracy scores of each model was obtained. Table shows the results of evaluation metrics of the research, through this the performance of the models can be analyzed. From which we can arrive at a conclusion that the best performance was given by Random Forest Classifier model, where an accuracy of 89 % was achieved after data optimization.

TABLE IV. OPTIMAL FEATURES IDENTIFIED IN SPSS

Parameters contributing towards PCOS		**Parameters contributing towards infertility**	
Feature	***Significance***	***Feature***	***Significance***
Cycle irregularity	0.000	No. of abortion	0.002
Cycle Length	0.000		
FSH,LH Ratio	0.006	Thickness of Endometrium	0.660
AMH	0.000		
Follicle no.	0.000	Vitamin D3	0.050
Follicle size	0.002		
BMI	0.000	AMH	0.002
Weight gain	0.000		

Therefore, it can be concluded that either biochemical profile alone or USG result alone can't serve as a diagnostic tool for the treating PCOS. Because both the factors that relate PCOS and infertility falls in both categories. AMH turns out to be a very promising feature to detect PCOS and infertility as per our results. Comparing the accuracies obtained for other studies in PCOS detection, 97% is the highest accuracy obtained in [1], 82% in [2], 93.9% in [3] and 90% in [5]. Our results are lesser than these, even though a direct comparison is meaningless. With optimization of weight parameters of classifiers, performance of the system might improve.

TABLE V. ACCURACY SCORE, SENSITIVITY, SPECIFICITY AND PRECISION OF VARIOUS MODELS

Algorithm Used	**Accuracy score**	**Sensitivity**	**Specificity**	**Precision**	**F1 score**
Logistic Regression (LR)	0.8536	0.6451	0.98039	0.952380	0.3845
K- Nearest Neighbors (KNN)	0.8658	0.8064	0.90196	0.83333	0.4098
Classification and Regression Trees (CART)	0.8292	0.8387	0.82352	0.74285	0.3939
Random Forest Classifier (RFC)	0.8902	0.7419	0.98039	0.95833	0.4182
Gaussian Naïve Bayes (NB)	0.8414	0.7419	0.90196	0.82142	0.3898
Support Vector Machines (SVM)	0.8292	0.5483	1.0	1.0	0.3541

VI. Conclusion

Polycystic Ovary Syndrome (PCOS) is one of the most common type of endocrine disorder in reproductive age women. This may result in infertility and anovulation. The diagnostic criterion includes the clinical and metabolic parameters which are biomarker for the disease. We developed a system that automates the PCOS detection based on minimal set of potential markers. Our methodology involves the formulation of a feature vector based on real time data from patients during clinical and radiological investigation while they visit a healthcare facility. The eight metabolic and Ultrasound image features identified with the PCA feature transformatin and statistical significance is found promising for discriminating between normal and PCOS patients. Among the various algorithms used, RF algorithm is found superior in performance. This automated system can act as an assistive tool for the doctor for saving considerable time in examining the patients and hence reducing the delay in diagnosing the risk of PCOS. Implications from the clinical expert survey for this work suggests that, innovations that uphold the medical ethics are always welcome in the field of medicine and healthcare. Researches that could bring out useful innovative methodologies like the effect of Vitamin D on PCOS, studies that put forth the impact of PCOS on preterm labor/abortions, attempt to unveil the number of lean PCOS patients etc. need to be held in future.

Acknowledgment

We thank the authorities of Sahrdaya CET, Thrissur, for providing financial support towards the successful completion of this work. We like to extend our gratitude towards the doctors (Dr. Krishnankutty, Kare Clinic, Thrissur, Dr. Neetha George, Jubilee Mission Hospital, Thrissur, Dr. Shyni Daniel, St. James Hospital, Chalakkudy, and Dr. Rahul, St. James Hospital, Chalakkudy) for their valuable sugestions and the permissions to conduct the data collection. We thank the former guide Mr Jis Paul and the project coordinator Ms. Annsonia M. Michael whose comments have greatly influenced the implementation of the work. Any errors found are our own, and should not tarnish the reputations of the esteemed persons.

References

[1] Cheng, J.J. and Mahalingaiah, S., 2018. Data mining and classification of polycystic ovaries in pelvic ultrasound reports. *bioRxiv*, p.254870.

[2] Dewi, R.M. and Wisesty, U.N., 2018, March. Classification of polycystic ovary based on ultrasound images using competitive neural network. In *Journal of Physics: Conference Series* (Vol. 971, No. 1, p. 012005). IOP Publishing.

[3] P. Mehrotra, J. Chatterjee, C. Chakraborty, B. Ghoshdastidar and S. Ghoshdastidar, "Automated screening of Polycystic Ovary Syndrome using machine learning techniques," *2011 Annual IEEE India Conference*, Hyderabad, 2011, pp. 1-5.

[4] Sachdeva, G., Gainder, S., Suri, V., Sachdeva, N. and Chopra, S., 2019. Obese and non-obese polycystic ovarian syndrome: Comparison of clinical, metabolic, hormonal parameters, and their differential response to clomiphene. *Indian journal of endocrinology and metabolism*, *23*(2), p.257.

[5] Zhang, X.Z., Pang, Y.L., Wang, X. and Li, Y.H., 2018. Computational characterization and identification of human polycystic ovary syndrome genes. *Scientific reports*, *8*(1), p.12949.

[6] Joham, A.E., Teede, H.J., Ranasinha, S., Zoungas, S. and Boyle, J., 2015. Prevalence of infertility and use of fertility treatment in women with polycystic ovary syndrome: data from a large community-based cohort study. *Journal of women's health*, *24*(4), pp.299-307.

[7] Dr. K. Meena, Dr. M. Manimekalai, S. Rethinavalli*, "A Literature Review on Polycystic Ovarian Syndrome and Data Mining Techniques",*Volume 4,Issue 12,December 2014,International Journal of Advanced Research in Computer Science and Software Engineering, ISSN: 2277 128X.

[8] Dhayat, N.A., Marti, N., Kollmann, Z., Troendle, A., Bally, L., Escher, G., Grössl, M., Ackermann, D., Ponte, B., Pruijm, M. and Müller, M., 2018. Urinary steroid profiling in women hints at a diagnostic signature of the polycystic ovary syndrome: A pilot study considering neglected steroid metabolites. *PloS one*, *13*(10), p.e0203903.

[9] Rotterdam EA-SPCWG. Revised 2003 consensus on diagnostic criteria and long-term health risks related to polycystic ovary syndrome. Fertil Steril. 2004;81(1):19-25.

[10] Dewailly, D., Lujan, M.E., Carmina, E., Cedars, M.I., Laven, J., Norman, R.J. and Escobar-Morreale, H.F., 2013. Definition and significance of polycystic ovarian morphology: a task force report from the Androgen Excess and Polycystic Ovary Syndrome Society. *Human reproduction update*, *20*(3), pp.334-352.

[11] Escobar-Morreale, H.F., 2018. Polycystic ovary syndrome: definition, aetiology, diagnosis and treatment. *Nature Reviews Endocrinology*, *14*(5), p.270.

[12] Lawrence, M.J., Eramian, M.G., Pierson, R.A. and Neufeld, E., 2007, May. Computer assisted detection of polycystic ovary morphology in ultrasound images. In *Fourth Canadian Conference on Computer and Robot Vision (CRV'07)* (pp. 105-112). IEEE.

[13] Dumesic, D.A., Oberfield, S.E., Stener-Victorin, E., Marshall, J.C., Laven, J.S. and Legro, R.S., 2015. Scientific statement on the diagnostic criteria, epidemiology, pathophysiology, and molecular genetics of polycystic ovary syndrome. *Endocrine reviews*, *36*(5), pp.487-525.

[14] Brower, M., Brennan, K., Pall, M. and Azziz, R., 2013. The severity of menstrual dysfunction as a predictor of insulin resistance in PCOS. The Journal of Clinical Endocrinology & Metabolism, 98(12), pp.E1967-E1971.

[15] Lawrence, M.J., Eramian, M.G., Pierson, R.A. and Neufeld, E., 2007, May. Computer assisted detection of polycystic ovary morphology in ultrasound images. In Fourth Canadian Conference on Computer and Robot Vision (CRV'07) (pp. 105-112). IEEE.

[16] McCartney, C.R. and Marshall, J.C., 2016. Polycystic ovary syndrome. *New England Journal of Medicine*, *375*(1), pp.54-64.

[17] Dumesic, D.A., Oberfield, S.E., Stener-Victorin, E., Marshall, J.C., Laven, J.S. and Legro, R.S., 2015. Scientific statement on the diagnostic criteria, epidemiology, pathophysiology, and molecular genetics of polycystic ovary syndrome. *Endocrine reviews*, *36*(5), pp.487-525.

[18] Norman, R.J., Dewailly, D., Legro, R.S. and Hickey, T.E., 2007. Polycystic ovary syndrome. *The Lancet*, *370*(9588), pp.685-697.

[19] Essah, P.A. and Nestler, J.E., 2006. The metabolic syndrome in polycystic ovary syndrome. *Journal of endocrinological investigation*, *29*(3), pp.270-280.

[20] Dhayat, N.A., Marti, N., Kollmann, Z., Troendle, A., Bally, L., Escher, G., Grössl, M., Ackermann, D., Ponte, B., Pruijm, M. and Müller, M., 2018. Urinary steroid profiling in women hints at a diagnostic signature of the polycystic ovary syndrome: A pilot study considering neglected steroid metabolites. PloS one, 13(10), p.e0203903.

[21] Gibson-Helm, M., Teede, H., Dunaif, A. and Dokras, A., 2016. Delayed diagnosis and a lack of information associated with dissatisfaction in women with polycystic ovary syndrome. *The Journal of Clinical Endocrinology & Metabolism*, *102*(2), pp.604-612.

DATA CLEANING & PRE-PROCESSING

PCOS_ML_Part_1

October 29, 2021

1 Data Cleaning And Data Preprocessing

This notebooks is written to clean our raw dataset and for solving unwanted issues so that we can use the data well structred dataset.

2 Importing Necessary Modules

[ ]:

```
import sys
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use("ggplot")
plt.rcParams['figure.figsize'] = (12, 8)
import seaborn as sns
sns.set(style='whitegrid', color_codes=True)
import warnings
warnings.filterwarnings('ignore')
```

3 Load Dataset

[ ]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:

```
df_f = pd.read_csv('/content/drive/MyDrive/PCOS/PCOS_data_without_infertility.
 ↪csv')
df_f.head(12).T
```

[ ]:

```
df_f.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541 entries, 0 to 540
Data columns (total 45 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Sl. No                     541 non-null    int64
 1   Patient File No.        541 non-null    int64
 2   PCOS (Y/N)              541 non-null    int64
 3    Age (yrs)              541 non-null    int64
 4   Weight (Kg)             541 non-null    float64
 5   Height(Cm)              541 non-null    float64
 6   BMI                     541 non-null    object
 7   Blood Group             541 non-null    int64
 8   Pulse rate(bpm)         541 non-null    int64
 9   RR (breaths/min)        541 non-null    int64
 10  Hb(g/dl)                541 non-null    float64
 11  Cycle(R/I)              541 non-null    int64
 12  Cycle length(days)      541 non-null    int64
 13  Marraige Status (Yrs)   540 non-null    float64
 14  Pregnant(Y/N)           541 non-null    int64
 15  No. of aborptions       541 non-null    int64
 16    I   beta-HCG(mIU/mL)  541 non-null    float64
 17  II    beta-HCG(mIU/mL)  541 non-null    object
 18  FSH(mIU/mL)             541 non-null    float64
 19  LH(mIU/mL)              541 non-null    float64
 20  FSH/LH                  541 non-null    object
 21  Hip(inch)               541 non-null    int64
 22  Waist(inch)             541 non-null    int64
 23  Waist:Hip Ratio         541 non-null    object
 24  TSH (mIU/L)             541 non-null    float64
 25  AMH(ng/mL)              541 non-null    object
 26  PRL(ng/mL)              541 non-null    float64
 27  Vit D3 (ng/mL)          541 non-null    float64
 28  PRG(ng/mL)              541 non-null    float64
 29  RBS(mg/dl)              541 non-null    float64
 30  Weight gain(Y/N)        541 non-null    int64
 31  hair growth(Y/N)        541 non-null    int64
 32  Skin darkening (Y/N)    541 non-null    int64
 33  Hair loss(Y/N)          541 non-null    int64
 34  Pimples(Y/N)            541 non-null    int64
 35  Fast food (Y/N)         540 non-null    float64
 36  Reg.Exercise(Y/N)       541 non-null    int64
 37  BP _Systolic (mmHg)     541 non-null    int64
 38  BP _Diastolic (mmHg)    541 non-null    int64
 39  Follicle No. (L)        541 non-null    int64
 40  Follicle No. (R)        541 non-null    int64
 41  Avg. F size (L) (mm)    541 non-null    float64
 42  Avg. F size (R) (mm)    541 non-null    float64
 43  Endometrium (mm)        541 non-null    float64
 44  Unnamed: 44             2 non-null      object
dtypes: float64(16), int64(23), object(6)
memory usage: 190.3+ KB
```

4 Explore Dataset

5 Check For Null Values

[ ]:
```
df_f.isnull()
```

[ ]:
```
     Sl. No  Patient File No.  …  Endometrium (mm)  Unnamed: 44
0     False             False  …             False         True
1     False             False  …             False         True
2     False             False  …             False         True
3     False             False  …             False         True
4     False             False  …             False         True
..      …               …  …               …          …
536   False             False  …             False         True
537   False             False  …             False         True
538   False             False  …             False         True
539   False             False  …             False         True
540   False             False  …             False         True

[541 rows x 45 columns]
```

[ ]:
```
df_f.isnull().sum()
```

[ ]:
```
Sl. No                      0
Patient File No.            0
PCOS (Y/N)                  0
 Age (yrs)                  0
Weight (Kg)                 0
Height(Cm)                  0
BMI                         0
Blood Group                 0
Pulse rate(bpm)             0
RR (breaths/min)            0
Hb(g/dl)                    0
Cycle(R/I)                  0
Cycle length(days)          0
Marraige Status (Yrs)       1
Pregnant(Y/N)               0
No. of aborptions           0
  I   beta-HCG(mIU/mL)      0
II    beta-HCG(mIU/mL)      0
FSH(mIU/mL)                 0
LH(mIU/mL)                  0
FSH/LH                      0
Hip(inch)                   0
Waist(inch)                 0
Waist:Hip Ratio             0
```

```
TSH (mIU/L)               0
AMH(ng/mL)                0
PRL(ng/mL)                0
Vit D3 (ng/mL)            0
PRG(ng/mL)                0
RBS(mg/dl)                0
Weight gain(Y/N)          0
hair growth(Y/N)          0
Skin darkening (Y/N)      0
Hair loss(Y/N)            0
Pimples(Y/N)              0
Fast food (Y/N)           1
Reg.Exercise(Y/N)         0
BP _Systolic (mmHg)       0
BP _Diastolic (mmHg)      0
Follicle No. (L)          0
Follicle No. (R)          0
Avg. F size (L) (mm)      0
Avg. F size (R) (mm)      0
Endometrium (mm)          0
Unnamed: 44             539
dtype: int64
```

[ ]:

```
# Drop the column containg almost all null and uninterpretable values
df_f = df_f.drop(columns='Unnamed: 44')
# Drop unnecessary columms
df_f = df_f.drop(columns=['Sl. No','Patient File No.'])
```

[ ]:

```
df_f.head(12).T
```

[ ]:

```
                           0       1       2  ...       9      10      11
PCOS (Y/N)                 0       0       1  ...       0       0       0
 Age (yrs)                28      36      33  ...      36      20      26
Weight (Kg)             44.6      65    68.8  ...      52      71      49
Height(Cm)               152   161.5     165  ...     150     163     160
BMI                     19.3  #NAME?  #NAME?  ...  #NAME?  #NAME?  #NAME?
Blood Group               15      15      11  ...      15      15      13
Pulse rate(bpm)           78      74      72  ...      80      80      72
RR (breaths/min)          22      20      18  ...      20      20      20
Hb(g/dl)               10.48    11.7    11.8  ...      10      10     9.5
Cycle(R/I)                 2       2       2  ...       4       2       2
Cycle length(days)         5       5       5  ...       2       5       5
Marraige Status (Yrs)      7      11      10  ...       4       4       3
Pregnant(Y/N)              0       1       1  ...       0       1       0
No. of aborptions          0       0       0  ...       0       2       1
  I   beta-HCG(mIU/mL)  1.99    60.8  494.08  ...    1.99  158.51    1.99
II    beta-HCG(mIU/mL)  1.99    1.99  494.08  ...    1.99  158.51    1.99
```

```
FSH(mIU/mL)              7.95    6.73    5.54  …     2.8    4.89    4.09
LH(mIU/mL)               3.68    1.09    0.88  …    1.51    2.02    1.47
FSH/LH                 #NAME?  #NAME?  #NAME?  …  #NAME?  #NAME?  #NAME?
Hip(inch)                  36      38      40  …      40      39      39
Waist(inch)                30      32      36  …      38      35      33
Waist:Hip Ratio        #NAME?  #NAME?  #NAME?  …  #NAME?  #NAME?  #NAME?
TSH (mIU/L)              0.68    3.16    2.54  …    6.65    1.56    3.98
AMH(ng/mL)               2.07    1.53    6.63  …    1.61    4.47    1.67
PRL(ng/mL)              45.16   20.09   10.52  …   11.74   13.47    21.1
Vit D3 (ng/mL)           17.1    61.3    49.7  …    27.7    18.1   29.18
PRG(ng/mL)               0.57    0.97    0.36  …    0.25    0.36    0.25
RBS(mg/dl)                 92      92      84  …     125     108     100
Weight gain(Y/N)            0       0       0  …       0       0       0
hair growth(Y/N)            0       0       0  …       0       0       0
Skin darkening (Y/N)        0       0       0  …       0       0       0
Hair loss(Y/N)              0       0       1  …       0       0       0
Pimples(Y/N)                0       0       1  …       0       0       0
Fast food (Y/N)             1       0       1  …       0       0       0
Reg.Exercise(Y/N)           0       0       0  …       0       0       0
BP _Systolic (mmHg)       110     120     120  …     110     110     120
BP _Diastolic (mmHg)       80      70      80  …      80      80      80
Follicle No. (L)            3       3      13  …       1       7       4
Follicle No. (R)            3       5      15  …       1      15       2
Avg. F size (L) (mm)       18      15      18  …      14      17      18
Avg. F size (R) (mm)       18      14      20  …      17      20      19
Endometrium (mm)          8.5     3.7      10  …     2.5       6     7.8

[42 rows x 12 columns]
```

6 Handling Missing Values

7 Drop Rows with null values

```
[ ]: df_f.dropna(axis=0, inplace=True)
```

```
[ ]: df_f.isnull().sum()
```

```
[ ]: PCOS (Y/N)              0
      Age (yrs)              0
     Weight (Kg)             0
     Height(Cm)              0
     BMI                     0
     Blood Group             0
     Pulse rate(bpm)         0
     RR (breaths/min)        0
     Hb(g/dl)                0
```

```
Cycle(R/I)                0
Cycle length(days)        0
Marraige Status (Yrs)     0
Pregnant(Y/N)             0
No. of aborptions         0
  I   beta-HCG(mIU/mL)    0
II    beta-HCG(mIU/mL)    0
FSH(mIU/mL)               0
LH(mIU/mL)                0
FSH/LH                    0
Hip(inch)                 0
Waist(inch)               0
Waist:Hip Ratio           0
TSH (mIU/L)               0
AMH(ng/mL)                0
PRL(ng/mL)                0
Vit D3 (ng/mL)            0
PRG(ng/mL)                0
RBS(mg/dl)                0
Weight gain(Y/N)          0
hair growth(Y/N)          0
Skin darkening (Y/N)      0
Hair loss(Y/N)            0
Pimples(Y/N)              0
Fast food (Y/N)           0
Reg.Exercise(Y/N)         0
BP _Systolic (mmHg)       0
BP _Diastolic (mmHg)      0
Follicle No. (L)          0
Follicle No. (R)          0
Avg. F size (L) (mm)      0
Avg. F size (R) (mm)      0
Endometrium (mm)          0
dtype: int64
```

```
[ ]: df_f.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 539 entries, 0 to 540
Data columns (total 42 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   PCOS (Y/N)              539 non-null    int64
 1    Age (yrs)              539 non-null    int64
 2   Weight (Kg)             539 non-null    float64
 3   Height(Cm)              539 non-null    float64
 4   BMI                     539 non-null    object
 5   Blood Group             539 non-null    int64
 6   Pulse rate(bpm)         539 non-null    int64
 7   RR (breaths/min)        539 non-null    int64
 8   Hb(g/dl)                539 non-null    float64
 9   Cycle(R/I)              539 non-null    int64
 10  Cycle length(days)      539 non-null    int64
 11  Marraige Status (Yrs)   539 non-null    float64
 12  Pregnant(Y/N)           539 non-null    int64
 13  No. of aborptions       539 non-null    int64
 14    I   beta-HCG(mIU/mL)  539 non-null    float64
 15  II    beta-HCG(mIU/mL)  539 non-null    object
 16  FSH(mIU/mL)             539 non-null    float64
 17  LH(mIU/mL)              539 non-null    float64
 18  FSH/LH                  539 non-null    object
 19  Hip(inch)               539 non-null    int64
 20  Waist(inch)             539 non-null    int64
 21  Waist:Hip Ratio         539 non-null    object
 22  TSH (mIU/L)             539 non-null    float64
 23  AMH(ng/mL)              539 non-null    object
 24  PRL(ng/mL)              539 non-null    float64
 25  Vit D3 (ng/mL)          539 non-null    float64
 26  PRG(ng/mL)              539 non-null    float64
 27  RBS(mg/dl)              539 non-null    float64
 28  Weight gain(Y/N)        539 non-null    int64
 29  hair growth(Y/N)        539 non-null    int64
 30  Skin darkening (Y/N)    539 non-null    int64
 31  Hair loss(Y/N)          539 non-null    int64
 32  Pimples(Y/N)            539 non-null    int64
 33  Fast food (Y/N)         539 non-null    float64
 34  Reg.Exercise(Y/N)       539 non-null    int64
 35  BP _Systolic (mmHg)     539 non-null    int64
 36  BP _Diastolic (mmHg)    539 non-null    int64
 37  Follicle No. (L)        539 non-null    int64
 38  Follicle No. (R)        539 non-null    int64
 39  Avg. F size (L) (mm)    539 non-null    float64
 40  Avg. F size (R) (mm)    539 non-null    float64
 41  Endometrium (mm)        539 non-null    float64
dtypes: float64(16), int64(21), object(5)
memory usage: 181.1+ KB
```

8 Solve corrupted values

```
[ ]: df_f
```

```
[ ]:      PCOS (Y/N)  Age (yrs)  …  Avg. F size (R) (mm)  Endometrium (mm)
     0             0         28  …                  18.0               8.5
     1             0         36  …                  14.0               3.7
     2             1         33  …                  20.0              10.0
```

```
3              0          37   …                   14.0              7.5
4              0          25   …                   14.0              7.0
..           …          …    …                   …                 …
536            0          35   …                   10.0              6.7
537            0          30   …                   18.0              8.2
538            0          36   …                    9.0              7.3
539            0          27   …                   16.0             11.5
540            1          23   …                   18.0              6.9

[539 rows x 42 columns]
```

```
[ ]: #calculating BMI
     df_f['BMI'] = (df_f['Weight (Kg)']/(df_f['Height(Cm) '])**2)*10000
     df_f
```

```
[ ]:      PCOS (Y/N)  Age (yrs)  …  Avg. F size (R) (mm)  Endometrium (mm)
     0              0         28  …                  18.0               8.5
     1              0         36  …                  14.0               3.7
     2              1         33  …                  20.0              10.0
     3              0         37  …                  14.0               7.5
     4              0         25  …                  14.0               7.0
     ..           …          …  …                   …                 …
     536            0         35  …                  10.0               6.7
     537            0         30  …                  18.0               8.2
     538            0         36  …                   9.0               7.3
     539            0         27  …                  16.0              11.5
     540            1         23  …                  18.0               6.9

     [539 rows x 42 columns]
```

```
[ ]: df_f.loc[:,"FSH/LH"] = df_f.loc[:,"FSH(mIU/mL)"] / df_f.loc[:,"LH(mIU/mL)"];
     df_f.loc[:,"FSH/LH"] = df_f.loc[:,"FSH/LH"].round(2);
```

```
[ ]: df_f.loc[:,"Waist:Hip Ratio"] = df_f.loc[:,"Waist(inch)"] / df_f.loc[:
      ↪,"Hip(inch)"]
     df_f.loc[:,"Waist:Hip Ratio"] = df_f.loc[:,"Waist:Hip Ratio"].round(2)
```

```
[ ]: df_f.head(12)
```

```
[ ]:    PCOS (Y/N)  Age (yrs)  …  Avg. F size (R) (mm)  Endometrium (mm)
     0            0         28  …                  18.0               8.5
     1            0         36  …                  14.0               3.7
     2            1         33  …                  20.0              10.0
     3            0         37  …                  14.0               7.5
     4            0         25  …                  14.0               7.0
     5            0         36  …                  20.0               8.0
     6            0         34  …                  16.0               6.8
```

```
7            0          33  …                      18.0                7.1
8            0          32  …                      17.0                4.2
9            0          36  …                      17.0                2.5
10           0          20  …                      20.0                6.0
11           0          26  …                      19.0                7.8

[12 rows x 42 columns]
```

[ ]:
```
df_f.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 539 entries, 0 to 540
Data columns (total 42 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   PCOS (Y/N)              539 non-null    int64
 1    Age (yrs)              539 non-null    int64
 2   Weight (Kg)             539 non-null    float64
 3   Height(Cm)              539 non-null    float64
 4   BMI                     539 non-null    float64
 5   Blood Group             539 non-null    int64
 6   Pulse rate(bpm)         539 non-null    int64
 7   RR (breaths/min)        539 non-null    int64
 8   Hb(g/dl)                539 non-null    float64
 9   Cycle(R/I)              539 non-null    int64
 10  Cycle length(days)      539 non-null    int64
 11  Marraige Status (Yrs)   539 non-null    float64
 12  Pregnant(Y/N)           539 non-null    int64
 13  No. of aborptions       539 non-null    int64
 14    I   beta-HCG(mIU/mL)  539 non-null    float64
 15  II    beta-HCG(mIU/mL)  539 non-null    object
 16  FSH(mIU/mL)             539 non-null    float64
 17  LH(mIU/mL)              539 non-null    float64
 18  FSH/LH                  539 non-null    float64
 19  Hip(inch)               539 non-null    int64
 20  Waist(inch)             539 non-null    int64
 21  Waist:Hip Ratio         539 non-null    float64
 22  TSH (mIU/L)             539 non-null    float64
 23  AMH(ng/mL)              539 non-null    object
 24  PRL(ng/mL)              539 non-null    float64
 25  Vit D3 (ng/mL)          539 non-null    float64
 26  PRG(ng/mL)              539 non-null    float64
 27  RBS(mg/dl)              539 non-null    float64
 28  Weight gain(Y/N)        539 non-null    int64
 29  hair growth(Y/N)        539 non-null    int64
 30  Skin darkening (Y/N)    539 non-null    int64
 31  Hair loss(Y/N)          539 non-null    int64
 32  Pimples(Y/N)            539 non-null    int64
 33  Fast food (Y/N)         539 non-null    float64
 34  Reg.Exercise(Y/N)       539 non-null    int64
 35  BP _Systolic (mmHg)     539 non-null    int64
 36  BP _Diastolic (mmHg)    539 non-null    int64
 37  Follicle No. (L)        539 non-null    int64
 38  Follicle No. (R)        539 non-null    int64
 39  Avg. F size (L) (mm)    539 non-null    float64
 40  Avg. F size (R) (mm)    539 non-null    float64
 41  Endometrium (mm)        539 non-null    float64
dtypes: float64(19), int64(21), object(2)
memory usage: 181.1+ KB
```

9 Replace Irrelevant Values

[ ]:

```
# df[df["Cycle(R/I)"] == 5]
df_f["Cycle(R/I)"].replace({5: 4}, inplace=True)
df_f["Cycle(R/I)"].replace({2: 0, 4: 1}, inplace=True)
```

[ ]:

```
df_f["II    beta-HCG(mIU/mL)"].replace({"1.99.": 1.99}, inplace=True)
```

[ ]:

```
df_f["II    beta-HCG(mIU/mL)"] = df_f["II    beta-HCG(mIU/mL)"].astype(float)
```

[ ]:

```
df_f[df_f["AMH(ng/mL)"]== "a"].T
```

[ ]:

```
                            305
PCOS (Y/N)                    0
 Age (yrs)                   37
Weight (Kg)                  56
Height(Cm)                  152
BMI                     24.2382
Blood Group                  13
Pulse rate(bpm)              74
RR (breaths/min)             20
Hb(g/dl)                   11.7
Cycle(R/I)                    0
Cycle length(days)            5
Marraige Status (Yrs)         9
Pregnant(Y/N)                 0
No. of aborptions             0
  I   beta-HCG(mIU/mL)       42
II    beta-HCG(mIU/mL)     1.99
FSH(mIU/mL)                2.91
LH(mIU/mL)                 0.35
FSH/LH                     8.31
Hip(inch)                    35
Waist(inch)                  33
Waist:Hip Ratio            0.94
```

```
TSH (mIU/L)                 16
AMH(ng/mL)                   a
PRL(ng/mL)                2.22
Vit D3 (ng/mL)            38.6
PRG(ng/mL)                 0.3
RBS(mg/dl)                 100
Weight gain(Y/N)             0
hair growth(Y/N)             0
Skin darkening (Y/N)         0
Hair loss(Y/N)               0
Pimples(Y/N)                 1
Fast food (Y/N)              0
Reg.Exercise(Y/N)            1
BP _Systolic (mmHg)        120
BP _Diastolic (mmHg)        70
Follicle No. (L)             4
Follicle No. (R)             5
Avg. F size (L) (mm)        17
Avg. F size (R) (mm)        16
Endometrium (mm)           5.6
```

```
[ ]: # df_f.drop(df_f["AMH(ng/mL)"]== "a", inplace=True)
     df_f.drop(df_f.loc[df_f["AMH(ng/mL)"]== "a"].index, inplace=True);
```

```
[ ]: df_f[df_f["AMH(ng/mL)"]== "a"]
```

```
[ ]: Empty DataFrame
     Columns: [PCOS (Y/N),  Age (yrs), Weight (Kg), Height(Cm) , BMI, Blood Group,
     Pulse rate(bpm) , RR (breaths/min), Hb(g/dl), Cycle(R/I), Cycle length(days),
     Marraige Status (Yrs), Pregnant(Y/N), No. of aborptions,   I   beta-HCG(mIU/mL),
     II    beta-HCG(mIU/mL), FSH(mIU/mL), LH(mIU/mL), FSH/LH, Hip(inch), Waist(inch),
     Waist:Hip Ratio, TSH (mIU/L), AMH(ng/mL), PRL(ng/mL), Vit D3 (ng/mL),
     PRG(ng/mL), RBS(mg/dl), Weight gain(Y/N), hair growth(Y/N), Skin darkening
     (Y/N), Hair loss(Y/N), Pimples(Y/N), Fast food (Y/N), Reg.Exercise(Y/N), BP
     _Systolic (mmHg), BP _Diastolic (mmHg), Follicle No. (L), Follicle No. (R), Avg.
     F size (L) (mm), Avg. F size (R) (mm), Endometrium (mm)]
     Index: []
```

```
[ ]: df_f["AMH(ng/mL)"] = df_f["AMH(ng/mL)"].astype(float)
```

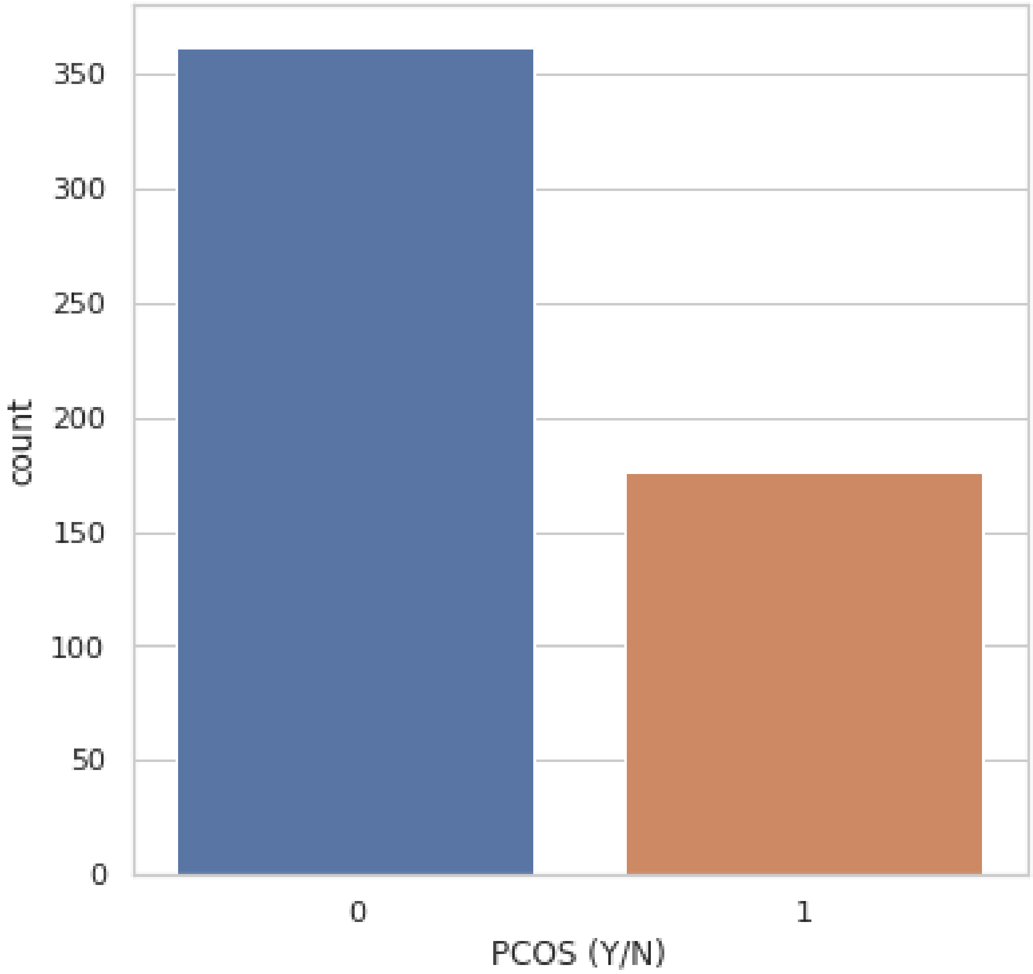
```
[ ]: df_f.info();
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 538 entries, 0 to 540
Data columns (total 42 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   PCOS (Y/N)              538 non-null    int64
 1     Age (yrs)              538 non-null    int64  
 2    Weight (Kg)             538 non-null    float64
 3    Height(Cm)              538 non-null    float64
 4    BMI                     538 non-null    float64
 5    Blood Group             538 non-null    int64  
 6    Pulse rate(bpm)         538 non-null    int64  
 7    RR (breaths/min)        538 non-null    int64  
 8    Hb(g/dl)                538 non-null    float64
 9    Cycle(R/I)              538 non-null    int64  
 10   Cycle length(days)      538 non-null    int64  
 11   Marraige Status (Yrs)   538 non-null    float64
 12   Pregnant(Y/N)           538 non-null    int64  
 13   No. of aborptions       538 non-null    int64  
 14     I   beta-HCG(mIU/mL)  538 non-null    float64
 15   II    beta-HCG(mIU/mL)  538 non-null    float64
 16   FSH(mIU/mL)             538 non-null    float64
 17   LH(mIU/mL)              538 non-null    float64
 18   FSH/LH                  538 non-null    float64
 19   Hip(inch)               538 non-null    int64  
 20   Waist(inch)             538 non-null    int64  
 21   Waist:Hip Ratio         538 non-null    float64
 22   TSH (mIU/L)             538 non-null    float64
 23   AMH(ng/mL)              538 non-null    float64
 24   PRL(ng/mL)              538 non-null    float64
 25   Vit D3 (ng/mL)          538 non-null    float64
 26   PRG(ng/mL)              538 non-null    float64
 27   RBS(mg/dl)              538 non-null    float64
 28   Weight gain(Y/N)        538 non-null    int64  
 29   hair growth(Y/N)        538 non-null    int64  
 30   Skin darkening (Y/N)    538 non-null    int64  
 31   Hair loss(Y/N)          538 non-null    int64  
 32   Pimples(Y/N)            538 non-null    int64  
 33   Fast food (Y/N)         538 non-null    float64
 34   Reg.Exercise(Y/N)       538 non-null    int64  
 35   BP _Systolic (mmHg)     538 non-null    int64  
 36   BP _Diastolic (mmHg)    538 non-null    int64  
 37   Follicle No. (L)        538 non-null    int64  
 38   Follicle No. (R)        538 non-null    int64  
 39   Avg. F size (L) (mm)    538 non-null    float64
 40   Avg. F size (R) (mm)    538 non-null    float64
 41   Endometrium (mm)        538 non-null    float64
dtypes: float64(21), int64(21)
memory usage: 180.7 KB
```

9.1 Count

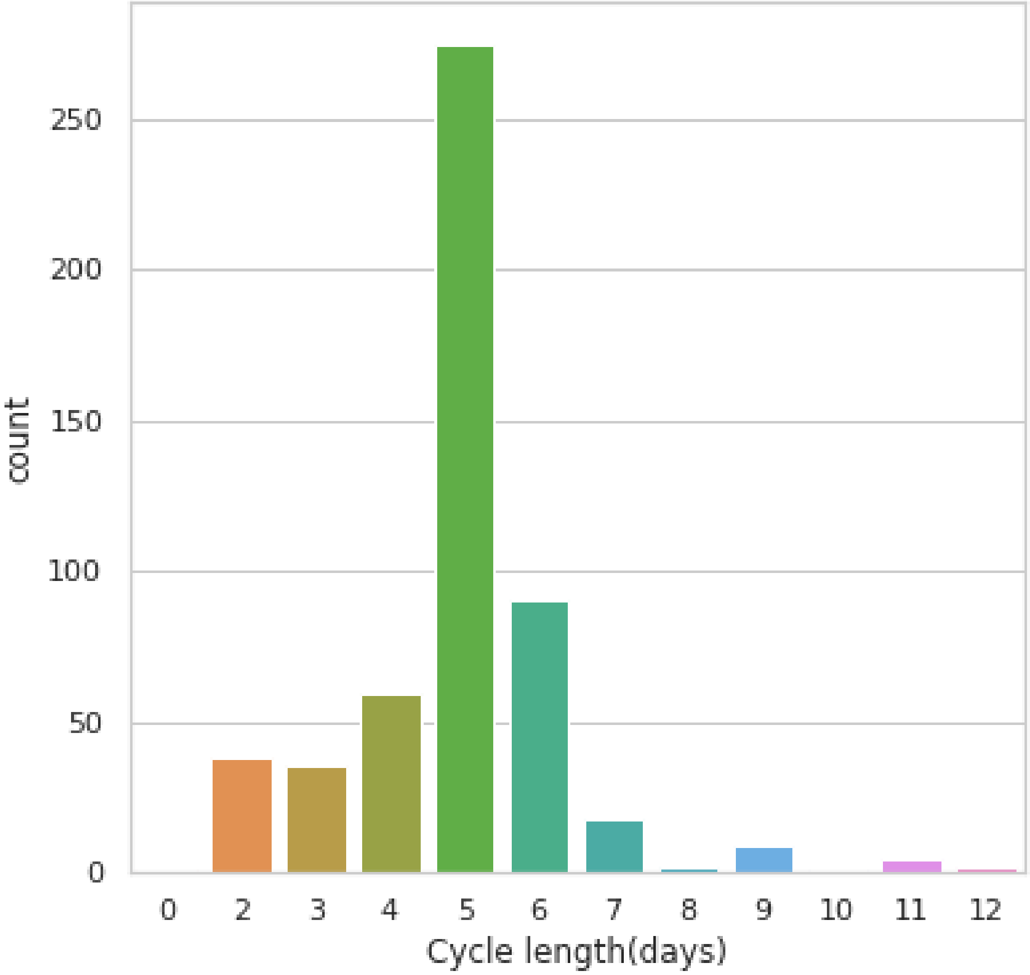
```
[ ]: print(df_f['PCOS (Y/N)'].value_counts())
     plt.figure(figsize=(6, 6))
     sns.countplot(
         x='PCOS (Y/N)',
         data=df_f
     );
```

```
0    362
1    176
Name: PCOS (Y/N), dtype: int64
```



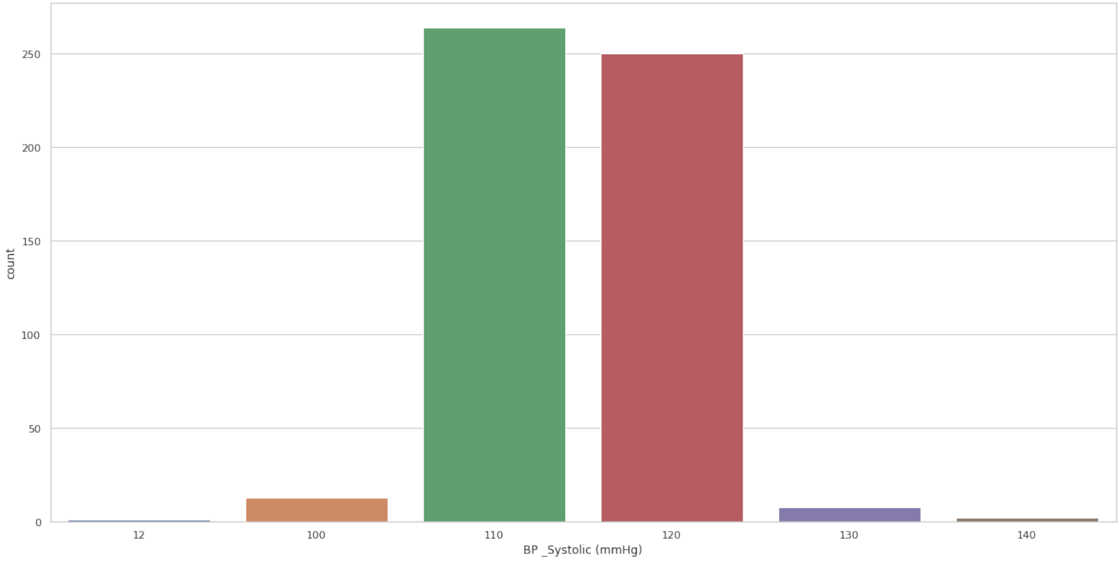
```
[ ]: print(df_f['Cycle length(days)'].value_counts())
     plt.figure(figsize=(6, 6))
     sns.countplot(
         x='Cycle length(days)',
         data=df_f
     );
```

```
5     275
6      91
4      60
2      38
3      36
7      18
9       9
11      5
12      2
8       2
10      1
0       1
Name: Cycle length(days), dtype: int64
```



```
[ ]: print(df_f['BP _Systolic (mmHg)'].value_counts())
     plt.figure(figsize=(20, 10))
     sns.countplot(
         x='BP _Systolic (mmHg)',
         data=df_f
     );
```

```
110    264
120    250
100     13
130      8
140      2
12       1
Name: BP _Systolic (mmHg), dtype: int64
```



```
[ ]: df_f["BP _Systolic (mmHg)"].replace({12: 120}, inplace=True)
     df_f["BP _Diastolic (mmHg)"].replace({8: 80}, inplace=True)
```

10 Save Cleaned Dataset

```
[ ]: #df_f.to_csv('../../datasets/PCOS_clean_data_without_infertility.csv',␣
      ↪index=False)
```

FEATURE SELECTION

PCOS_ML_Part_2

October 29, 2021

1 Feature Selection For The Dataset

1.1 Importing Libraries

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
df=pd.read_csv("/content/drive/MyDrive/PCOS/PCOS_clean_data_without_infertility.
 ↪csv")
```

[ ]:
```
df.describe().T
```

[ ]:
```
                        count        mean  …       75%       max
PCOS (Y/N)              538.0    0.327138  …    1.0000      1.00
Age (yrs)               538.0   31.420074  …   35.0000     48.00
Weight (Kg)             538.0   59.644052  …   65.0000    108.00
Height(Cm)              538.0  156.480104  …  160.0000    180.00
BMI                     538.0   24.323625  …   26.6625     38.90
Blood Group             538.0   13.802974  …   15.0000     18.00
Pulse rate(bpm)         538.0   73.250929  …   74.0000     82.00
RR (breaths/min)        538.0   19.236059  …   20.0000     28.00
Hb(g/dl)                538.0   11.160558  …   11.7750     14.80
Cycle(R/I)              538.0    0.276952  …    1.0000      1.00
Cycle length(days)      538.0    4.938662  …    5.0000     12.00
Marraige Status (Yrs)   538.0    7.683457  …   10.0000     30.00
Pregnant(Y/N)           538.0    0.382900  …    1.0000      1.00
No. of aborptions       538.0    0.289963  …    0.0000      5.00
I   beta-HCG(mIU/mL)    538.0  667.215271  …  297.0500  32460.97
II    beta-HCG(mIU/mL)  538.0  239.550333  …   99.1750  25000.00
FSH(mIU/mL)             538.0   14.669054  …    6.4175   5052.00
LH(mIU/mL)              538.0    6.503933  …    3.6800   2018.00
FSH/LH                  538.0    6.900279  …    3.8700   1372.83
Hip(inch)               538.0   37.998141  …   40.0000     48.00
Waist(inch)             538.0   33.840149  …   36.0000     47.00
Waist:Hip Ratio         538.0    0.891245  …    0.9300      0.98
TSH (mIU/L)             538.0    2.960935  …    3.5700     65.00
AMH(ng/mL)              538.0    5.623035  …    6.9750     66.00
```

```
PRL(ng/mL)             538.0   24.393216  …   29.9500    128.24
Vit D3 (ng/mL)         538.0   50.036781  …   34.4750   6014.66
PRG(ng/mL)             538.0    0.612660  …    0.4575     85.00
RBS(mg/dl)             538.0   99.864684  …  107.0000    350.00
Weight gain(Y/N)       538.0    0.379182  …    1.0000      1.00
hair growth(Y/N)       538.0    0.275093  …    1.0000      1.00
Skin darkening (Y/N)   538.0    0.306691  …    1.0000      1.00
Hair loss(Y/N)         538.0    0.453532  …    1.0000      1.00
Pimples(Y/N)           538.0    0.490706  …    1.0000      1.00
Fast food (Y/N)        538.0    0.516729  …    1.0000      1.00
Reg.Exercise(Y/N)      538.0    0.245353  …    0.0000      1.00
BP _Systolic (mmHg)    538.0  114.832714  …  120.0000    140.00
BP _Diastolic (mmHg)   538.0   77.081784  …   80.0000    100.00
Follicle No. (L)       538.0    6.120818  …    9.0000     22.00
Follicle No. (R)       538.0    6.646840  …   10.0000     20.00
Avg. F size (L) (mm)   538.0   15.014498  …   18.0000     24.00
Avg. F size (R) (mm)   538.0   15.448643  …   18.0000     24.00
Endometrium (mm)       538.0    8.477454  …    9.8000     18.00

[42 rows x 8 columns]
```

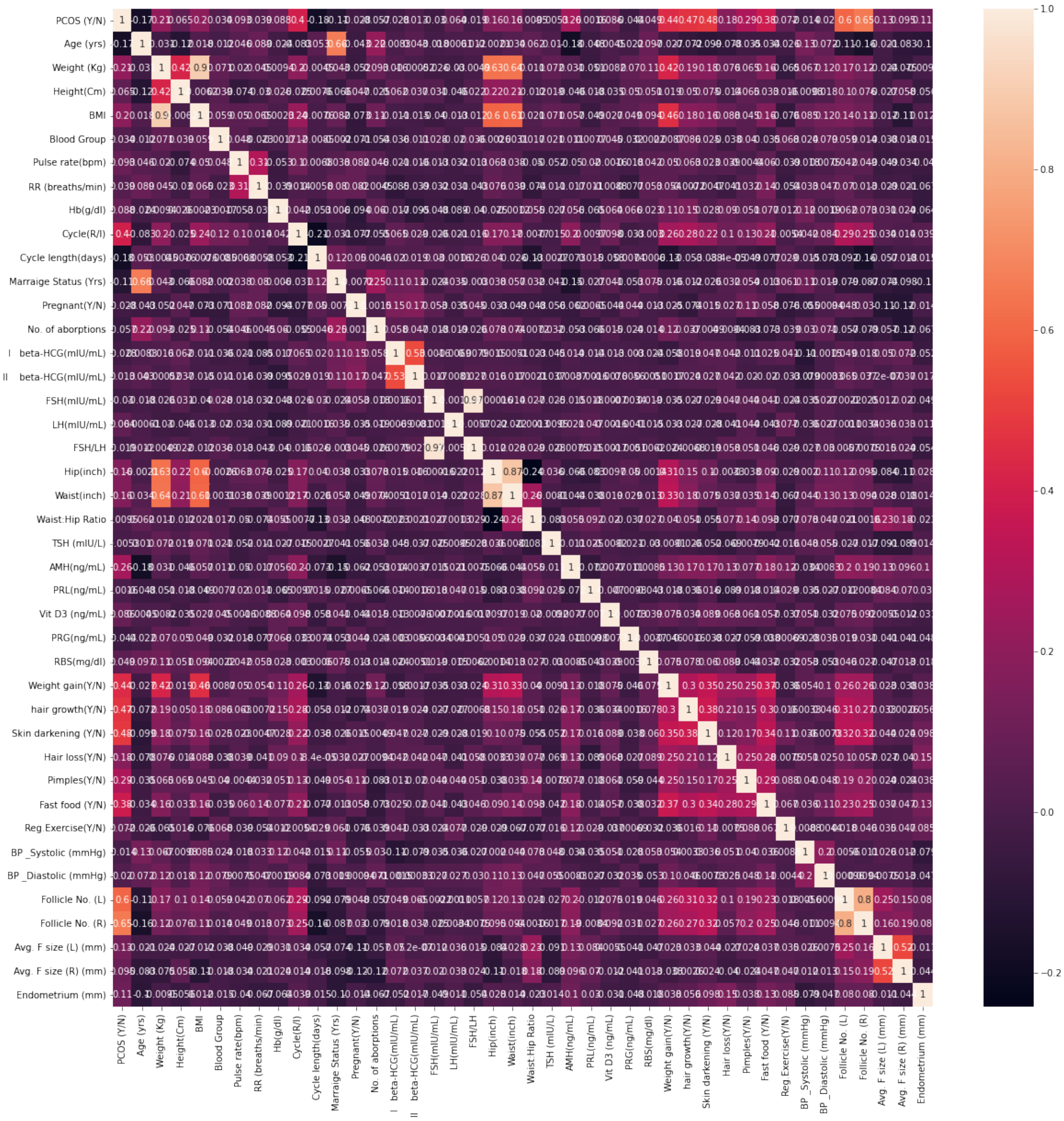
```
[ ]: df.corr()
```

```
[ ]:                        PCOS (Y/N)  …  Endometrium (mm)
     PCOS (Y/N)               1.000000  …          0.107639
     Age (yrs)               -0.171349  …         -0.100115
     Weight (Kg)              0.209969  …         -0.009452
     Height(Cm)               0.065465  …         -0.056273
     BMI                      0.199302  …          0.012331
     Blood Group              0.033701  …         -0.015257
     Pulse rate(bpm)          0.092699  …         -0.040456
     RR (breaths/min)         0.038641  …         -0.066551
     Hb(g/dl)                 0.087809  …         -0.063592
     Cycle(R/I)               0.400668  …          0.039000
     Cycle length(days)      -0.183811  …         -0.014630
     Marraige Status (Yrs)   -0.113406  …         -0.104838
     Pregnant(Y/N)           -0.027632  …         -0.013675
     No. of aborptions       -0.057316  …         -0.067136
     I   beta-HCG(mIU/mL)    -0.028074  …         -0.051580
     II    beta-HCG(mIU/mL)   0.012808  …          0.017295
     FSH(mIU/mL)             -0.030384  …         -0.049132
     LH(mIU/mL)               0.064074  …          0.010917
     FSH/LH                  -0.018512  …         -0.053760
     Hip(inch)                0.160882  …          0.028465
     Waist(inch)              0.161922  …          0.014308
     Waist:Hip Ratio          0.009505  …         -0.023082
     TSH (mIU/L)             -0.005277  …          0.014302
```

```
AMH(ng/mL)                0.263974  …          0.104496
PRL(ng/mL)                0.001649  …          0.030021
Vit D3 (ng/mL)            0.085825  …         -0.031307
PRG(ng/mL)               -0.043897  …         -0.047946
RBS(mg/dl)                0.049452  …         -0.017851
Weight gain(Y/N)          0.443093  …          0.038255
hair growth(Y/N)          0.466508  …          0.055972
Skin darkening (Y/N)      0.481323  …          0.097781
Hair loss(Y/N)            0.176496  …          0.151038
Pimples(Y/N)              0.290335  …          0.038164
Fast food (Y/N)           0.380985  …          0.130807
Reg.Exercise(Y/N)         0.071979  …          0.084927
BP _Systolic (mmHg)      -0.013765  …         -0.079425
BP _Diastolic (mmHg)      0.019870  …         -0.047305
Follicle No. (L)          0.601208  …          0.080313
Follicle No. (R)          0.650929  …          0.079607
Avg. F size (L) (mm)      0.129997  …         -0.011326
Avg. F size (R) (mm)      0.094528  …         -0.043720
Endometrium (mm)          0.107639  …          1.000000

[42 rows x 42 columns]
```

[ ]:

```
plt.figure(figsize=(20,20))
sns.heatmap(df.corr(), annot=True)
plt.show()
```



1.2 Feature Selcetion Based On Correlation

The Pearson Method is used to choose the features for model building. The method implemented includes choosing of only those features whose correlation i.e corr_features is greater than 0.4 while discarding all the others.

This gave us the following features for final model building in subsequent notebooks. - Cycle(R/I) - Weight gain(Y/N) - hair growth(Y/N) - Skin darkening (Y/N) - Follicle No. (L) - Follicle No. (R) - PCOS (Y/N)—> Dependent Variable

```
[ ]: corr_features=df.corrwith(df["PCOS (Y/N)"],method='pearson').abs().
      ↪sort_values(ascending=True)
```

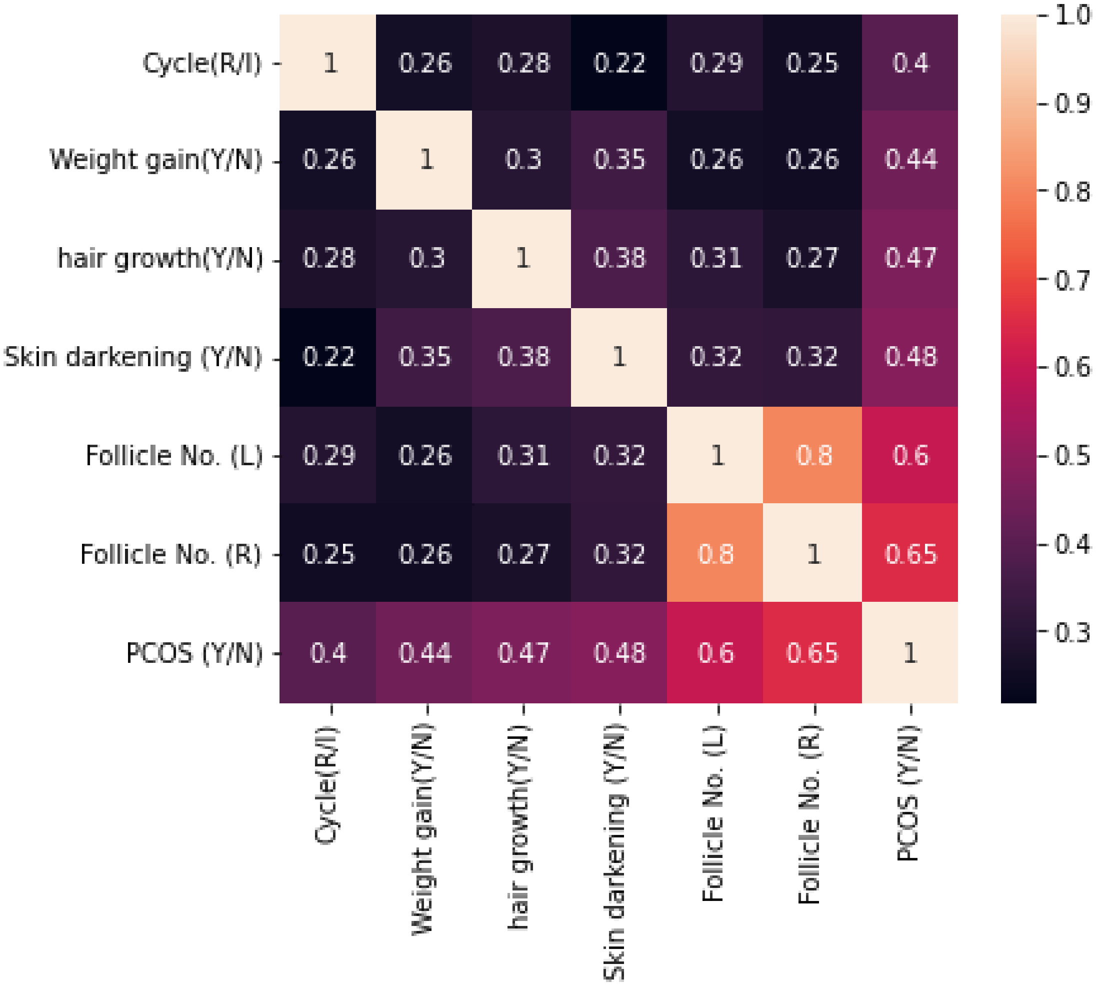
```
corr_features=corr_features[corr_features>0.4].index
corr_features
```

```
[ ]: Index(['Cycle(R/I)', 'Weight gain(Y/N)', 'hair growth(Y/N)',
            'Skin darkening (Y/N)', 'Follicle No. (L)', 'Follicle No. (R)',
            'PCOS (Y/N)'],
           dtype='object')
```

```
[ ]: df_f=df[corr_features]
     df_f.head()
```

```
[ ]:    Cycle(R/I)  Weight gain(Y/N)  …  Follicle No. (R)  PCOS (Y/N)
     0           0                 0  …                 3           0
     1           0                 0  …                 5           0
     2           0                 0  …                15           1
     3           0                 0  …                 2           0
     4           0                 0  …                 4           0

     [5 rows x 7 columns]
```

```
[ ]: plt.figure(figsize=(6,5))
     sns.heatmap(df_f.corr(), annot=True)
     plt.show()
```


Cycle(R/I)
Weight gain(Y/N)
hair growth(Y/N)
Skin darkening (Y/N)
Follicle No. (L)
Follicle No. (R)
PCOS (Y/N)
1 0.26 0.28 0.22 0.29 0.25 0.4
0.26 1 0.3 0.35 0.26 0.26 0.44
0.28 0.3 1 0.38 0.31 0.27 0.47
0.22 0.35 0.38 1 0.32 0.32 0.48
0.29 0.26 0.31 0.32 1 0.8 0.6
0.25 0.26 0.27 0.32 0.8 1 0.65
0.4 0.44 0.47 0.48 0.6 0.65 1
Cycle(R/I)
Weight gain(Y/N)
hair growth(Y/N)
Skin darkening (Y/N)
Follicle No. (L)
Follicle No. (R)
PCOS (Y/N)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3

LOGISTIC REGRESSION & NAÏVE BAYES

PCOS_ML_Part_3

October 29, 2021

1 Algorithm: Logistic Regression & Naive Bayes

1.1 Importing Libraries

[ ]:

```
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

import pickle
```

[ ]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:

```
df = pd.read_csv("/content/drive/MyDrive/PCOS/clean_data.csv")
```

[ ]:

```
df.shape
```

[ ]:

```
(534, 7)
```

[ ]:

```
X = df.drop(columns=["PCOS (Y/N)"])
X.head()
```

[ ]:

```
   Follicle No. (R)  Follicle No. (L)  …  Weight gain(Y/N)  Cycle(R/I)
0                 3                 3  …                 0           2
1                 5                 3  …                 0           2
2                15                13  …                 0           2
3                 2                 2  …                 0           2
4                 4                 3  …                 0           2

[5 rows x 6 columns]
```

```
[ ]: y = df["PCOS (Y/N)"].values
```

1.2 Splitting Dataset

```
[ ]: X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,␣
      ↪random_state=12)
```

2 Logistic Regression Implementation

```
[ ]: lr = LogisticRegression()
     lr.fit(X_train, y_train)
```

```
[ ]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                        intercept_scaling=1, l1_ratio=None, max_iter=100,
                        multi_class='auto', n_jobs=None, penalty='l2',
                        random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                        warm_start=False)
```

```
[ ]: print(f"Score in Train Data : {lr.score(X_train,y_train)}")
```

```
Score in Train Data : 0.9016393442622951
```

```
[ ]: y_pred=lr.predict(X_test)
```

```
[ ]: print("Logistic Regression model accuracy(in %):", metrics.
      ↪accuracy_score(y_test, y_pred)*100)
```

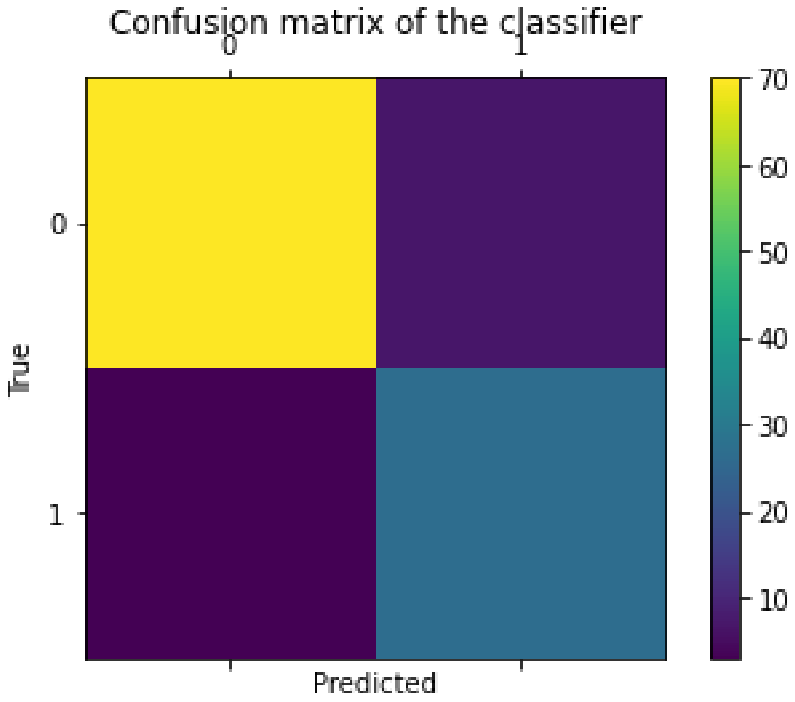
```
Logistic Regression model accuracy(in %): 90.65420560747664
```

```
[ ]:  print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.91      0.93        77
           1       0.79      0.90      0.84        30

    accuracy                           0.91       107
   macro avg       0.88      0.90      0.89       107
weighted avg       0.91      0.91      0.91       107
```

```
[ ]: cm = confusion_matrix(y_test, y_pred)
     fig = plt.figure()

     ax = fig.add_subplot(111)
     cax = ax.matshow(cm)
     plt.title('Confusion matrix of the classifier')
     fig.colorbar(cax)
```

```
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```



[ ]:
```
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
print("True Negatives: ",tn)
print("False Positives: ",fp)
print("False Negatives: ",fn)
print("True Positives: ",tp)
specificity = tn / (tn+fp)
print(specificity)
```

```
True Negatives:  70
False Positives:  7
False Negatives:  3
True Positives:  27
0.9090909090909091
```

[ ]:
```
sensitivity=tp/(fn+tp)
print(sensitivity)
```

```
0.9
```

2.1 Naive Bayes Implementation

```
[ ]: from sklearn.naive_bayes import GaussianNB
```

```
[ ]: gnb = GaussianNB()
     gnb.fit(X_train, y_train)
```

```
[ ]: GaussianNB(priors=None, var_smoothing=1e-09)
```

```
[ ]: y_pred = gnb.predict(X_test)
     print("Gaussian Naive Bayes model accuracy(in %):", metrics.
      ↪accuracy_score(y_test, y_pred)*100)
```

```
Gaussian Naive Bayes model accuracy(in %): 87.85046728971963
```

```
[ ]: tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
     print("True Negatives: ",tn)
     print("False Positives: ",fp)
     print("False Negatives: ",fn)
     print("True Positives: ",tp)
     specificity = tn / (tn+fp)
     print(specificity)
```

```
True Negatives:  66
False Positives:  11
False Negatives:  2
True Positives:  28
0.8571428571428571
```

```
[ ]: sensitivity=tp/(fn+tp)
     print(sensitivity)
```

```
0.9333333333333333
```

```
[ ]:  print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.86      0.91        77
           1       0.72      0.93      0.81        30

    accuracy                           0.88       107
   macro avg       0.84      0.90      0.86       107
weighted avg       0.90      0.88      0.88       107
```

RANDOM FOREST CLASSIFIER (RFC)

RFC Final

October 29, 2021

1 Algorithm: Random Forest Classifier(RFC)

[19]:
```
import pandas as pd
```

[20]:
```
df=pd.read_csv("/content/drive/MyDrive/PCOS/clean_data.csv")
```

[21]:
```
X = df.drop(['PCOS (Y/N)'], axis=1)

y = df['PCOS (Y/N)']
```

2 Splitting Dataset

[22]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 0)
```

[23]:
```
X_train.shape, X_test.shape
```

[23]:
```
((427, 6), (107, 6))
```

[24]:
```
cols = X_train.columns
```

2.1 Importing Necessary Libraries

[33]:
```
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn import model_selection
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

[34]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,␣
 ↪random_state=23)
```

2.2 RFC Implementation

[39]:

```
rfc = RandomForestClassifier(n_estimators=1000)

rfc.fit(X_train,y_train)
# predictions
rfc_predict = rfc.predict(X_test)
```

[40]:

```
print("Random forest model accuracy(in %):", metrics.accuracy_score(y_test,␣
 ↪rfc_predict)*100)
```

```
Random forest model accuracy(in %): 89.83050847457628
```

[43]:

```
tn, fp, fn, tp = confusion_matrix(y_test, rfc_predict).ravel()
print("True Negatives: ",tn)
print("False Positives: ",fp)
print("False Negatives: ",fn)
print("True Positives: ",tp)
specificity = tn / (tn+fp)
print(specificity)
```

```
True Negatives:  112
False Positives:  5
False Negatives:  13
True Positives:  47
0.9572649572649573
```

[44]:

```
sensitivity=tp/(fn+tp)
print(sensitivity)
```

```
0.7833333333333333
```

[41]:

```
print(classification_report(y_test, rfc_predict))
```

```
              precision    recall  f1-score   support

           0       0.90      0.96      0.93       117
           1       0.90      0.78      0.84        60

    accuracy                           0.90       177
   macro avg       0.90      0.87      0.88       177
weighted avg       0.90      0.90      0.90       177
```

K-NEAREST NEIGHBORS (KNN)

KNN

October 29, 2021

1 Algorithm: K Nearest Neighbour (KNN)

1.1 Importing Libraries

[1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

[2]:

```
df = pd.read_csv('/content/drive/MyDrive/clean_data.csv')
```

[3]:

```
X = df.drop(columns=["PCOS (Y/N)"])
y = df["PCOS (Y/N)"].values
```

1.2 Feature Scaling

[4]:

```
scaler = StandardScaler()
scaler.fit(df.drop('PCOS (Y/N)', axis=1))
scaled_features = scaler.transform(df.drop('PCOS (Y/N)', axis=1))
scaled_data = pd.DataFrame(scaled_features, columns = df.drop('PCOS (Y/N)',␣
 ↪axis=1).columns)
```

[5]:

```
x = scaled_data
y = df['PCOS (Y/N)']
```

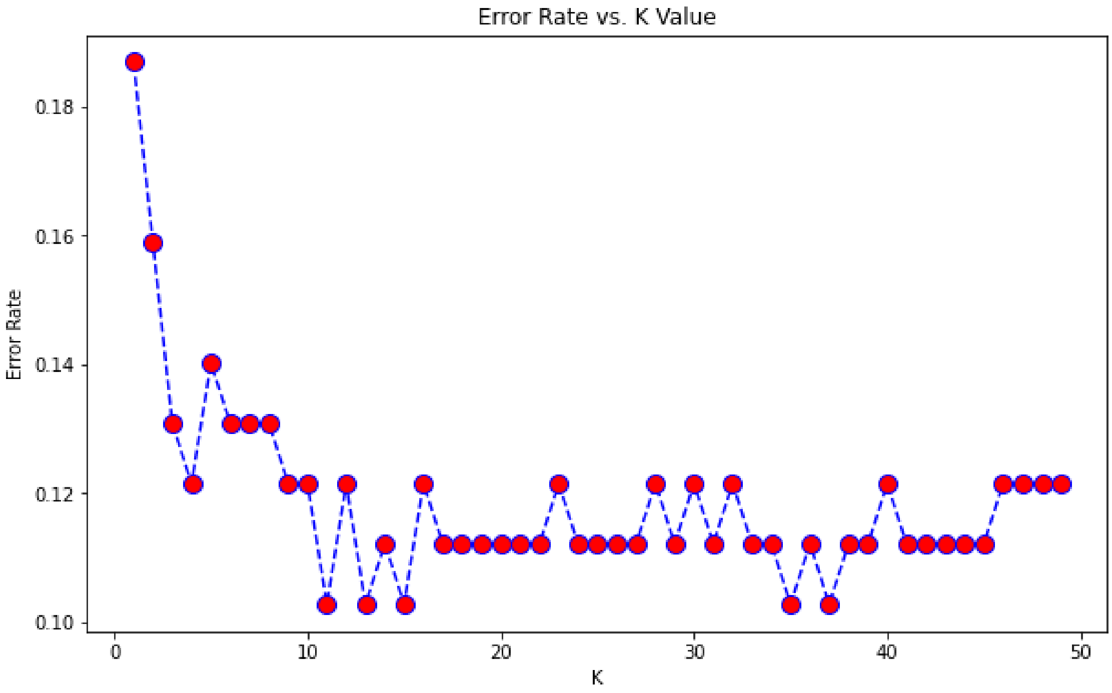
1.3 Splitting Dataset

```
[6]: x_training_data, x_test_data, y_training_data, y_test_data =␣
      ↪train_test_split(x, y, test_size = 0.2,random_state=42)
```

1.4 KNN Implementation

```
[7]: #Selecting an optimal K value

     error_rates = []
     for i in np.arange(1, 50):
         new_model = KNeighborsClassifier(n_neighbors = i)
         new_model.fit(x_training_data, y_training_data)
         new_predictions = new_model.predict(x_test_data)
         error_rates.append(np.mean(new_predictions != y_test_data))

     plt.figure(figsize=(10,6))
     plt.plot(range(1,50),error_rates,color='blue', linestyle='dashed',
              marker='o',markerfacecolor='red', markersize=10)
     plt.title('Error Rate vs. K Value')
     plt.xlabel('K')
     plt.ylabel('Error Rate')
     print("Minimum error:-",min(error_rates),"at K =",error_rates.
      ↪index(min(error_rates)))
```

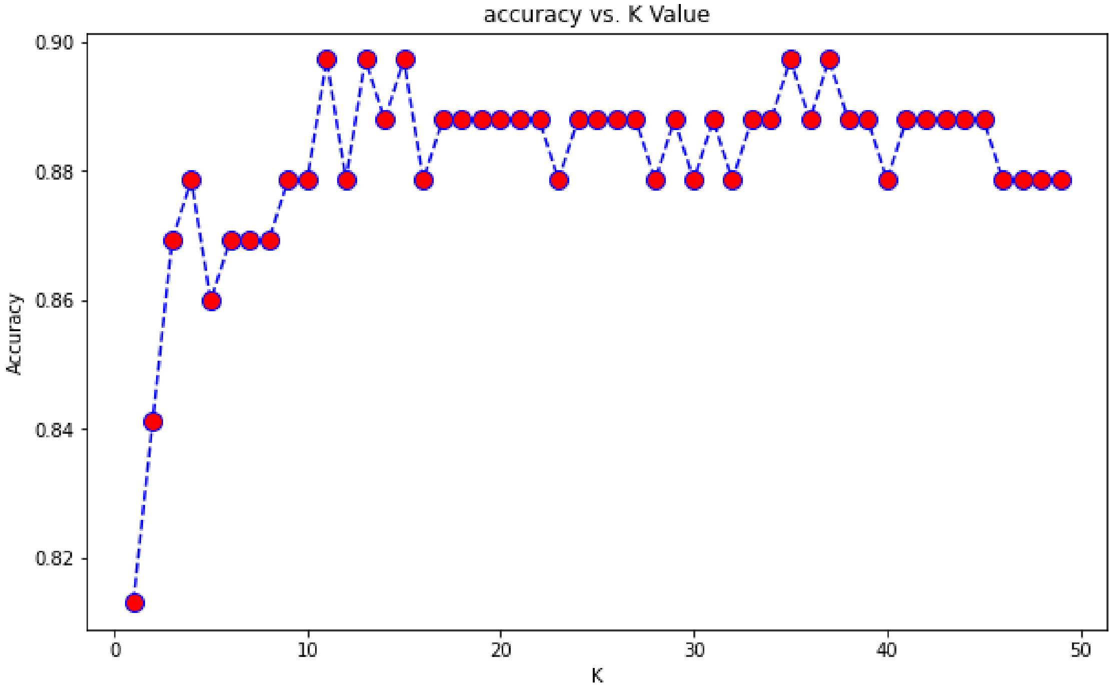
```
Minimum error:- 0.102803738317757 at K = 10
```



[19]:

```
acc = []

for i in range(1,50):
    neigh = KNeighborsClassifier(n_neighbors = i).fit(x_training_data,␣
 ↪y_training_data)
    yhat = neigh.predict(x_test_data)
    acc.append(metrics.accuracy_score(y_test_data, yhat))

plt.figure(figsize=(10,6))
plt.plot(range(1,50),acc,color = 'blue',linestyle='dashed',
         marker='o',markerfacecolor='red', markersize=10)
plt.title('accuracy vs. K Value')
plt.xlabel('K')
plt.ylabel('Accuracy')
print("Maximum training accuracy:-",max(acc),"at K =",acc.index(max(acc)))
```

```
Maximum training accuracy:- 0.897196261682243 at K = 10
```



[13]:

```
neigh = KNeighborsClassifier(n_neighbors = 10).fit(x_training_data,␣
 ↪y_training_data)
predictions = neigh.predict(x_test_data)
```

[20]:
```
print("K Nearest Neighbour model for K = 10 testing accuracy(in %):", metrics.
 ↪accuracy_score(y_test_data, predictions)*100)
```

```
K Nearest Neighbour model for K = 10 testing accuracy(in %): 87.85046728971963
```

[15]:
```
print(classification_report(y_test_data, predictions))
```

```
              precision    recall  f1-score   support

           0       0.88      0.96      0.92        75
           1       0.88      0.69      0.77        32

    accuracy                           0.88       107
   macro avg       0.88      0.82      0.84       107
weighted avg       0.88      0.88      0.87       107
```

[16]:
```
tn, fp, fn, tp = confusion_matrix(y_test_data, predictions).ravel()
print("True Negatives: ",tn)
print("False Positives: ",fp)
print("False Negatives: ",fn)
print("True Positives: ",tp)
```

```
True Negatives:  72
False Positives:  3
False Negatives:  10
True Positives:  22
```

[17]:
```
specificity = tn / (tn+fp)
print(specificity)
```

```
0.96
```

[18]:
```
sensitivity=tp/(fn+tp)
print(sensitivity)
```

```
0.6875
```

SUPPORT VECTOR MACHINES (SVM)

SVM_PCOS (1)

October 29, 2021

1 Algorithm: Support Vector Machines

1.1 Importing Libraries

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
df=pd.read_csv("/content/drive/MyDrive/PCOS/clean_data.csv")
```

[4]:
```
X = df.drop(['PCOS (Y/N)'], axis=1)

y = df['PCOS (Y/N)']
```

[5]:
```
X.shape
```

[5]:
```
(534, 6)
```

[6]:
```
y.shape
```

[6]:
```
(534,)
```

1.2 Splitting Dataset

[7]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 0)
```

[8]:
```
# check the shape of X_train and X_test

X_train.shape, X_test.shape
```

[8]:
```
((427, 6), (107, 6))
```

[9]:
```
cols = X_train.columns
```

1.3 Feature Scaling

[10]:

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

[11]:

```
X_train = pd.DataFrame(X_train, columns=[cols])
```

[12]:

```
X_test = pd.DataFrame(X_test, columns=[cols])
```

1.4 SVM Implementation

[13]:

```
# 12. Run SVM with default hyperparameters

#Default hyperparameter means C=1.0, kernel=rbf and gamma=auto among other␣
 ↪parameters.

# import SVC classifier
from sklearn.svm import SVC


# import metrics to compute accuracy
from sklearn.metrics import accuracy_score


# instantiate classifier with default hyperparameters
svc=SVC()


# fit classifier to training set
svc.fit(X_train,y_train)


# make predictions on test set
y_pred=svc.predict(X_test)


# compute and print accuracy score
print('Model accuracy score with default hyperparameters: {0:0.4f}'.␣
 ↪format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score with default hyperparameters: 0.8879
```

1.5 Running SVM with Linear Kernel

Linear kernel is used when the data is linearly separable. It means that data can be separated using a single line. It is one of the most common kernels to be used. It is mostly used when there are large number of features in a dataset. Linear kernel is often used for text classification purposes.

Training with a linear kernel is usually faster, because we only need to optimize the C regularization parameter. When training with other kernels, we also need to optimize the parameter. So, performing a grid search will usually take more time.

[14]:
```
# Run SVM with linear kernel and C=1.0
# instantiate classifier with linear kernel and C=1.0
linear_svc=SVC(kernel='linear', C=1.0)


# fit classifier to training set
linear_svc.fit(X_train,y_train)


# make predictions on test set
y_pred_test=linear_svc.predict(X_test)


# compute and print accuracy score
print('Model accuracy score with linear kernel and C=1.0 : {0:0.4f}'.␣
 ↪format(accuracy_score(y_test, y_pred_test)))
```

```
Model accuracy score with linear kernel and C=1.0 : 0.8972
```

1.6 Checking for Overfitting or Underfitting

[15]:
```
print('Training set score: {:.4f}'.format(linear_svc.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(linear_svc.score(X_test, y_test)))
```

```
Training set score: 0.9157
Test set score: 0.8972
```

[ ]:
```
# The training-set accuracy score is 0.9157 while the test-set accuracy to be 0.
 ↪8972.
# These two values are quite comparable. So, there is no question of␣
 ↪overfitting.
```

1.7 Running SVM with Polynomial Kernel

Polynomial kernel represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables. The polynomial kernel looks not only at the given features of input samples to determine their similarity, but also combinations of the input samples.

[16]:

```
# Run SVM with polynomial kernel and C=1.0
# instantiate classifier with polynomial kernel and C=1.0
poly_svc=SVC(kernel='poly', C=1.0)


# fit classifier to training set
poly_svc.fit(X_train,y_train)


# make predictions on test set
y_pred=poly_svc.predict(X_test)


# compute and print accuracy score
print('Model accuracy score with polynomial kernel and C=1.0 : {0:0.4f}'.␣
 ↪format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score with polynomial kernel and C=1.0 : 0.9065
```

[17]:

```
print('Training set score: {:.4f}'.format(poly_svc.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(poly_svc.score(X_test, y_test)))
```

```
Training set score: 0.9321
Test set score: 0.9065
```

[19]:

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

print('Confusion matrix\n\n', cm)

print('\nTrue Positives(TP) = ', cm[0,0])

print('\nTrue Negatives(TN) = ', cm[1,1])

print('\nFalse Positives(FP) = ', cm[0,1])

print('\nFalse Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix

 [[69  2]
 [ 8 28]]

True Positives(TP) =  69

True Negatives(TN) =  28
```

```
False Positives(FP) =  2

False Negatives(FN) =  8
```

[21]:

```python
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.97      0.93        71
           1       0.93      0.78      0.85        36

    accuracy                           0.91       107
   macro avg       0.91      0.87      0.89       107
weighted avg       0.91      0.91      0.90       107
```

[23]:

```python
sensitivity=cm[0,0]/(cm[1,0]+cm[0,0])
print(sensitivity)
```

```
0.8961038961038961
```

[25]:

```python
specificity = cm[1,1] / (cm[1,1]+cm[0,1])
print(specificity)
```

```
0.9333333333333333
```

1.8 Running SVM with Sigmoid Kernel

Sigmoid kernel has its origin in neural networks. We can use it as the proxy for neural networks.

[ ]:

```python
# Run SVM with sigmoid kernel and C=1.0
# instantiate classifier with sigmoid kernel and C=1.0
sigmoid_svc=SVC(kernel='sigmoid', C=1.0)


# fit classifier to training set
sigmoid_svc.fit(X_train,y_train)


# make predictions on test set
y_pred=sigmoid_svc.predict(X_test)


# compute and print accuracy score
print('Model accuracy score with sigmoid kernel and C=1.0 : {0:0.4f}'.␣
 ↪format(accuracy_score(y_test, y_pred)))
```

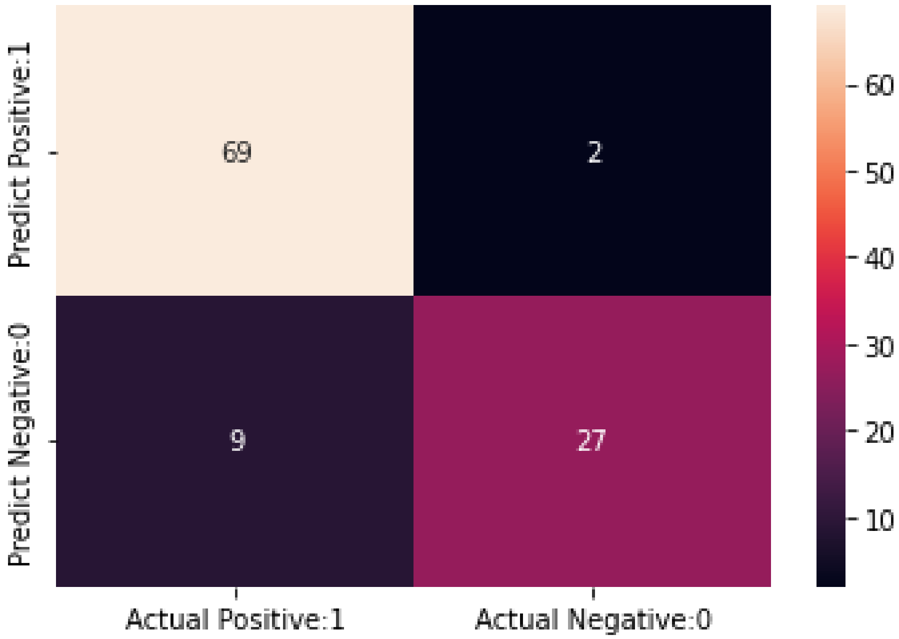
```
Model accuracy score with sigmoid kernel and C=1.0 : 0.8692
```

[ ]:
```python
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred_test)

print('Confusion matrix\n\n', cm)

print('\nTrue Positives(TP) = ', cm[0,0])

print('\nTrue Negatives(TN) = ', cm[1,1])

print('\nFalse Positives(FP) = ', cm[0,1])

print('\nFalse Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix

 [[69  2]
 [ 9 27]]

True Positives(TP) =  69

True Negatives(TN) =  27

False Positives(FP) =  2

False Negatives(FN) =  9
```

[ ]:
```python
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual␣
 ↪Negative:0'],
                                  index=['Predict Positive:1', 'Predict Negative:
 ↪0'])

sns.heatmap(cm_matrix, annot=True)
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7ac552c850>
```



```
[ ]: # Classification report for Poly Kernel
     from sklearn.metrics import classification_report

     print(classification_report(y_test, y_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.88      0.97      0.93        71
           1       0.93      0.75      0.83        36

    accuracy                           0.90       107
   macro avg       0.91      0.86      0.88       107
weighted avg       0.90      0.90      0.89       107
```

XGBOOST

Xgboost_PCOS

October 29, 2021

1 Algorithm: eXtreme Gradient Boosting(XGBoost)

1.1 Importing Libraries

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from numpy import loadtxt
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

[2]:
```
df = pd.read_csv('/content/drive/MyDrive/PCOS/clean_data.csv')
df.shape
```

[2]:
```
(534, 7)
```

[3]:
```
X = df.drop(columns=["PCOS (Y/N)"])
y = df["PCOS (Y/N)"].values
```

1.2 Splitting Dataset

[38]:
```
seed = 13
test_size = 0.2
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size,␣
 ↪random_state=seed)
```

1.3 Feature Scaling

[39]:
```
# Feature Scaling
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
```

```
X_test = scaler.transform(X_test)
```

1.4 XGBoost Implementation

XGBoost (eXtreme Gradient Boosting) is an advanced implementation of gradient boosting algorithm.

The overall parameters have been divided into 3 categories by XGBoost authors:

- General Parameters: Guide the overall functioning
- Booster Parameters: Guide the individual booster (tree/regression) at each step
- Learning Task Parameters: Guide the optimization performed

```
[40]: model = XGBClassifier(learning_rate =0.1, n_estimators=1000, max_depth=5,
      min_child_weight=1, gamma=0, subsample=0.8,
      colsample_bytree=0.8, objective= 'binary:logistic',
      nthread=4, scale_pos_weight=1, seed=27)
      model.fit(X_train, y_train)
```

```
[40]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                    colsample_bynode=1, colsample_bytree=0.8, gamma=0,
                    learning_rate=0.1, max_delta_step=0, max_depth=5,
                    min_child_weight=1, missing=None, n_estimators=1000, n_jobs=1,
                    nthread=4, objective='binary:logistic', random_state=0,
                    reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=27,
                    silent=None, subsample=0.8, verbosity=1)
```

1. General Paramenters:

booster [default=gbtree] Select the type of model to run at each iteration. It has 2 options: gbtree: tree-based models gblinear: linear models

silent [default=0]: Silent mode is activated is set to 1, i.e. no running messages will be printed. It's generally good to keep it 0 as the messages might help in understanding the model.

nthread [default to maximum number of threads available if not set] This is used for parallel processing and number of cores in the system should be entered If you wish to run on all cores, value should not be entered and algorithm will detect automatically

2. Booster Parameters

eta [default=0.3] Analogous to learning rate in GBM Makes the model more robust by shrinking the weights on each step Typical final values to be used: 0.01-0.2

min_child_weight [default=1] Defines the minimum sum of weights of all observations required in a child. This is similar to min_child_leaf in GBM but not exactly. This refers to min "sum of weights" of observations while GBM has min "number of observations". Used to control over-fitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree. Too high values can lead to under-fitting hence, it should be tuned using CV.

max_depth [default=6] The maximum depth of a tree, same as GBM. Used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample. Should be tuned using CV. Typical values: 3-10 max_leaf_nodes The maximum number of terminal nodes or leaves in a tree. Can be defined in place of max_depth. Since binary trees are created, a depth of ‘n’ would produce a maximum of 2^n leaves. If this is defined, GBM will ignore max_depth.

gamma [default=0] A node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split. Makes the algorithm conservative. The values can vary depending on the loss function and should be tuned.

max_delta_step [default=0] In maximum delta step we allow each tree’s weight estimation to be. If the value is set to 0, it means there is no constraint. If it is set to a positive value, it can help making the update step more conservative. Usually this parameter is not needed, but it might help in logistic regression when class is extremely imbalanced. This is generally not used but you can explore further if you wish.

subsample [default=1] Same as the subsample of GBM. Denotes the fraction of observations to be randomly samples for each tree. Lower values make the algorithm more conservative and prevents overfitting but too small values might lead to under-fitting. Typical values: 0.5-1

colsample_bytree [default=1] Similar to max_features in GBM. Denotes the fraction of columns to be randomly samples for each tree. Typical values: 0.5-1

colsample_bylevel [default=1] Denotes the subsample ratio of columns for each split, in each level. don’t use this often because subsample and colsample_bytree will do the job for you. but you can explore further if you feel so.

lambda [default=1] L2 regularization term on weights (analogous to Ridge regression) This used to handle the regularization part of XGBoost. Though many data scientists don’t use it often, it should be explored to reduce overfitting.

alpha [default=0] L1 regularization term on weight (analogous to Lasso regression) Can be used in case of very high dimensionality so that the algorithm runs faster when implemented scale_pos_weight [default=1] A value greater than 0 should be used in case of high class imbalance as it helps in faster convergence.

3. Learning Task Parameters

objective [default=reg:linear] This defines the loss function to be minimized. Mostly used values are: binary:logistic –logistic regression for binary classification, returns predicted probability (not class) multi:softmax –multiclass classification using the softmax objective, returns predicted class (not probabilities) you also need to set an additional num_class (number of classes) parameter defining the number of unique classes multi:softprob –same as softmax, but returns predicted probability of each data point belonging to each class.

eval_metric [default according to objective] The metric to be used for validation data. The default values are rmse for regression and error for classification. Typical values are: rmse – root mean square error mae – mean absolute error logloss – negative log-likelihood error – Binary classification error rate (0.5 threshold) merror – Multiclass classification error rate mlogloss – Multiclass logloss auc: Area under the curve

seed [default=0] The random number seed. Can be used for generating reproducible results and also for parameter tuning.

[41]:

```
y_pred = model.predict(X_test)
predictions = [round(value) for value in y_pred]
```

[42]:

```
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 90.65%
```

[43]:

```
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
print("True Negatives: ",tn)
print("False Positives: ",fp)
print("False Negatives: ",fn)
print("True Positives: ",tp)
specificity = tn / (tn+fp)
print(specificity)
```

```
True Negatives:  68
False Positives:  6
False Negatives:  4
True Positives:  29
0.918918918918919
```

[45]:

```
sensitivity=tp/(fn+tp)
print(sensitivity)
```

```
0.8787878787878788
```

[44]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.92      0.93        74
           1       0.83      0.88      0.85        33

    accuracy                           0.91       107
   macro avg       0.89      0.90      0.89       107
weighted avg       0.91      0.91      0.91       107
```

MULTI-LAYER PERCEPTRON (MLP CLASSIFIER)

MLP_PCOS

October 29, 2021

1 Algorithm: Multilayer Perceptron Classifier (MLP)

A multilayer perceptron is a type of feed-forward artificial neural network that generates a set of outputs from a set of inputs

MLPClassifier trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters.

It can also have a regularization term added to the loss function that shrinks model parameters to prevent overfitting.

This implementation works with data represented as dense numpy arrays or sparse scipy arrays of floating point values.

[23]:
```
import pandas as pd
```

[24]:
```
df=pd.read_csv("/content/drive/MyDrive/PCOS/clean_data.csv")
```

[25]:
```
X = df.drop(['PCOS (Y/N)'], axis=1)

y = df['PCOS (Y/N)']
```

1.1 Splitting Data

[26]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 0)
```

[27]:
```
X_train.shape, X_test.shape
```

[27]:
```
((427, 6), (107, 6))
```

[28]:
```
cols = X_train.columns
```

1.2 Feature Scaling

[29]:

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

[30]:

```
X_train = pd.DataFrame(X_train, columns=[cols])
```

[31]:

```
X_test = pd.DataFrame(X_test, columns=[cols])
```

[38]:

```
def test_results(model, X_test, y_test):
    from sklearn.metrics import confusion_matrix
    y_pred = model.predict(X_test)
    tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()

    accuracy = (tp + tn)/(tp + fp + tn + fn)
    print("Accuracy: ", '{:.2f}'.format(accuracy * 100))
    print("True Negative:", tn)
    print("True Positve:", tp)
    print("False Positive:", fp)
    print("False Negative:", fn)
    print()
    print("-------------------------------------------------------")
    print("Negative Class Results")
    precision = (tp / (tp + fp))
    recall =  (tp  / (tp + fn))
    f1_score = (2 * (precision * recall) / (precision + recall))
    print("Precision (N): ", '{:.2f}'.format(precision * 100))
    print("Recall (N): ", '{:.2f}'.format(recall * 100))
    print("F1 Score (N):" ,  '{:.2f}'.format(f1_score * 100))
    print()
    print("-------------------------------------------------------")
    print("Positive Class Results")
    precision = (tn / (tn + fn))
    recall =  (tn  / (tn + fp))
    f1_score = (2 * (precision * recall) / (precision + recall))
    print("Precision (P): ", '{:.2f}'.format(precision * 100))
    print("Recall (P): ", '{:.2f}'.format(recall * 100))
    print("F1 Score (P):" , '{:.2f}'.format(f1_score * 100))
    print("Specificity:")
    specificity = tn / (tn+fp)
    print(specificity)
    print("Sensitivity:")
    sensitivity=tp/(fn+tp)
    print(sensitivity)
```

1.3 Importing MLP Classifier From SKLearn

We can relate this model with the Keras Sequential Model.

[33]:

```
from sklearn.neural_network import MLPClassifier
```

[34]:

```
mlp = MLPClassifier(hidden_layer_sizes=(60, 60), activation='tanh',␣
 ↪solver='adam',
                    alpha=0.0001, learning_rate='adaptive',␣
 ↪learning_rate_init=0.001,
                    power_t=0.5, max_iter=200, shuffle=False, random_state=0,
                    tol=0.0001, verbose=False, warm_start=False, momentum=0.9,
                    nesterovs_momentum=True, early_stopping=False,␣
 ↪validation_fraction=0.1,
                    beta_1=0.9, beta_2=0.999, epsilon=1e-08,␣
 ↪n_iter_no_change=10, max_fun=15000)
```

Parameters:

hidden_layer_sizestuple, length = n_layers - 2, default=(100,) The ith element represents the number of neurons in the ith hidden layer.

activation{'identity', 'logistic', 'tanh', 'relu'}, default='relu' Activation function for the hidden layer.

'identity', no-op activation, useful to implement linear bottleneck, returns f(x) = x

'logistic', the logistic sigmoid function, returns f(x) = 1 / (1 + exp(-x)).

'tanh', the hyperbolic tan function, returns f(x) = tanh(x).

'relu', the rectified linear unit function, returns f(x) = max(0, x)

solver{'lbfgs', 'sgd', 'adam'}, default='adam' The solver for weight optimization.

'lbfgs' is an optimizer in the family of quasi-Newton methods.

'sgd' refers to stochastic gradient descent.

'adam' refers to a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba

Note: The default solver 'adam' works pretty well on relatively large datasets (with thousands of training samples or more) in terms of both training time and validation score. For small datasets, however, 'lbfgs' can converge faster and perform better.

alphafloat, default=0.0001 L2 penalty (regularization term) parameter.

learning_rate{'constant', 'invscaling', 'adaptive'}, default='constant' Learning rate schedule for weight updates.

‘constant’ is a constant learning rate given by ‘learning_rate_init’.

‘invscaling’ gradually decreases the learning rate at each time step ‘t’ using an inverse scaling exponent of ‘power_t’. effective_learning_rate = learning_rate_init / pow(t, power_t)

‘adaptive’ keeps the learning rate constant to ‘learning_rate_init’ as long as training loss keeps decreasing. Each time two consecutive epochs fail to decrease training loss by at least tol, or fail to increase validation score by at least tol if ‘early_stopping’ is on, the current learning rate is divided by 5.

Only used when solver=‘sgd’.

learning_rate_initdouble, default=0.001 The initial learning rate used. It controls the step-size in updating the weights. Only used when solver=’sgd’ or ‘adam’.

power_tdouble, default=0.5 The exponent for inverse scaling learning rate. It is used in updating effective learning rate when the learning_rate is set to ‘invscaling’. Only used when solver=’sgd’.

max_iterint, default=200 Maximum number of iterations. The solver iterates until convergence (determined by ‘tol’) or this number of iterations. For stochastic solvers (‘sgd’, ‘adam’), note that this determines the number of epochs (how many times each data point will be used), not the number of gradient steps.

nesterovs_momentumbool, default=True Whether to use Nesterov’s momentum. Only used when solver=’sgd’ and momentum > 0.

validation_fractionfloat, default=0.1 The proportion of training data to set aside as validation set for early stopping. Must be between 0 and 1. Only used if early_stopping is True.

beta_1float, default=0.9 Exponential decay rate for estimates of first moment vector in adam, should be in [0, 1). Only used when solver=’adam’.

beta_2float, default=0.999 Exponential decay rate for estimates of second moment vector in adam, should be in [0, 1). Only used when solver=’adam’.

epsilonfloat, default=1e-8 Value for numerical stability in adam. Only used when solver=’adam’.

n_iter_no_changeint, default=10 Maximum number of epochs to not meet tol improvement. Only effective when solver=’sgd’ or ‘adam’.

[35]:

```
mlp.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:571:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[35]:

```
MLPClassifier(activation='tanh', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(60, 60), learning_rate='adaptive',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=0, shuffle=False, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

[39]:
```
test_results(mlp, X_test, y_test)
```

```
Accuracy:  92.52
True Negative: 69
True Positve: 30
False Positive: 2
False Negative: 6

-------------------------------------------------------
Negative Class Results
Precision (N):  93.75
Recall (N):  83.33
F1 Score (N): 88.24

-------------------------------------------------------
Positive Class Results
Precision (P):  92.00
Recall (P):  97.18
F1 Score (P): 94.52
Specificity:
0.971830985915493
Sensitivity:
0.8333333333333334
```

1.4 Trying Activation Function To Be ReLu

[46]:
```
mlp1 = MLPClassifier(hidden_layer_sizes=(60, 60), activation='relu',␣
 ↪solver='adam',
                     alpha=0.0001, learning_rate='adaptive',␣
 ↪learning_rate_init=0.001,
                     power_t=0.5, max_iter=200, shuffle=False, random_state=0,
                     tol=0.0001, verbose=False, warm_start=False, momentum=0.9,
                     nesterovs_momentum=True, early_stopping=False,␣
 ↪validation_fraction=0.1,
                     beta_1=0.9, beta_2=0.999, epsilon=1e-08,␣
 ↪n_iter_no_change=10, max_fun=15000)
```

[47]:
```
mlp1.fit(X_train, y_train)
```

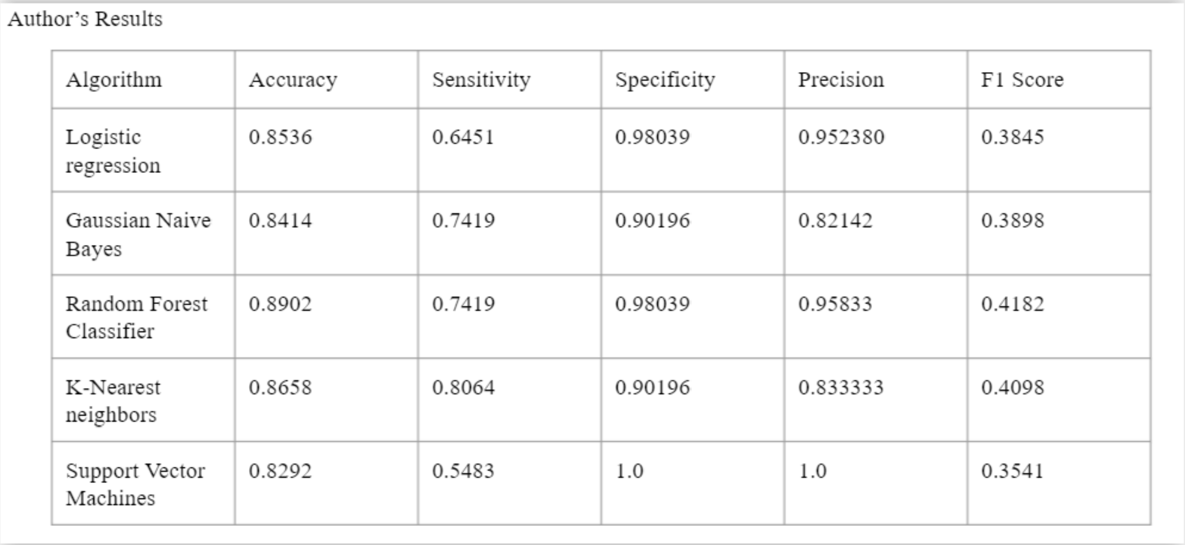
```
/usr/local/lib/python3.7/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:571:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
[47]: MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
                    beta_2=0.999, early_stopping=False, epsilon=1e-08,
                    hidden_layer_sizes=(60, 60), learning_rate='adaptive',
                    learning_rate_init=0.001, max_fun=15000, max_iter=200,
                    momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
                    power_t=0.5, random_state=0, shuffle=False, solver='adam',
                    tol=0.0001, validation_fraction=0.1, verbose=False,
                    warm_start=False)
```

```
[48]: test_results(mlp1, X_test, y_test)
```

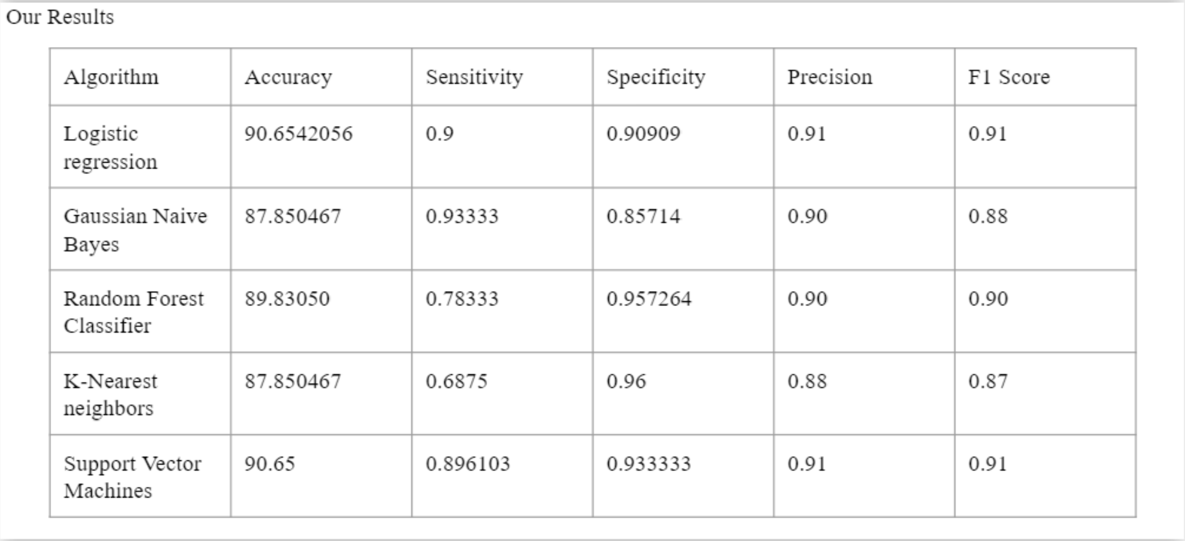
```
Accuracy:  87.85
True Negative: 67
True Positve: 27
False Positive: 4
False Negative: 9

-------------------------------------------------------
Negative Class Results
Precision (N):  87.10
Recall (N):  75.00
F1 Score (N): 80.60

-------------------------------------------------------
Positive Class Results
Precision (P):  88.16
Recall (P):  94.37
F1 Score (P): 91.16
Specificity:
0.9436619718309859
Sensitivity:
0.75
```

RESULTS



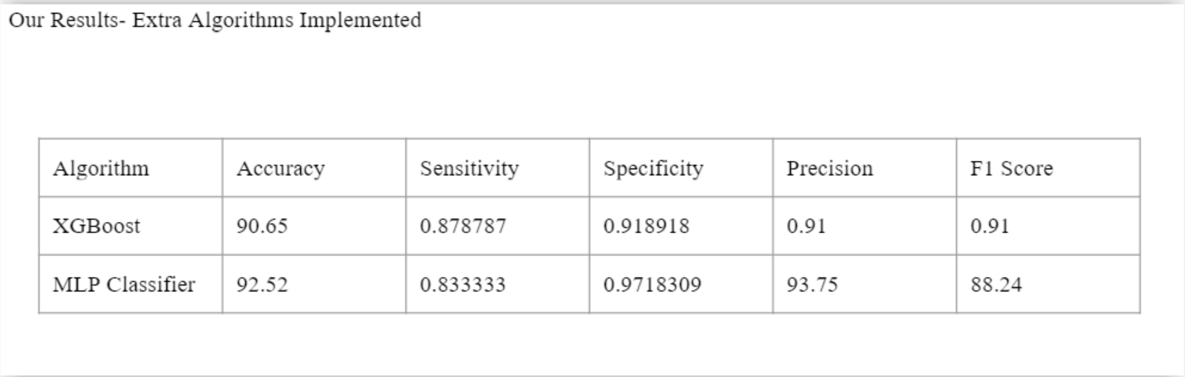
Author’s Results

Algorithm	Accuracy	Sensitivity	Specificity	Precision	F1 Score
Logistic regression	0.8536	0.6451	0.98039	0.952380	0.3845
Gaussian Naive Bayes	0.8414	0.7419	0.90196	0.82142	0.3898
Random Forest Classifier	0.8902	0.7419	0.98039	0.95833	0.4182
K-Nearest neighbors	0.8658	0.8064	0.90196	0.833333	0.4098
Support Vector Machines	0.8292	0.5483	1.0	1.0	0.3541



Our Results

Algorithm	Accuracy	Sensitivity	Specificity	Precision	F1 Score
Logistic regression	90.6542056	0.9	0.90909	0.91	0.91
Gaussian Naive Bayes	87.850467	0.93333	0.85714	0.90	0.88
Random Forest Classifier	89.83050	0.78333	0.957264	0.90	0.90
K-Nearest neighbors	87.850467	0.6875	0.96	0.88	0.87
Support Vector Machines	90.65	0.896103	0.933333	0.91	0.91



Our Results- Extra Algorithms Implemented

Algorithm	Accuracy	Sensitivity	Specificity	Precision	F1 Score
XGBoost	90.65	0.878787	0.918918	0.91	0.91
MLP Classifier	92.52	0.833333	0.9718309	93.75	88.24

Changes and modifications made:

- Our dataset is slightly different as some columns such as 'Diabetes' are missing.
- Where their dataset has a single parameter, ours has the same one split in two. For example, Follicle No. in theirs, Follicle No. (L) and Follicle No. (R) in ours
- The features we have extracted are through EDA by checking the heatmap and correlation matrix. They have extracted features mainly through SPSS software and through discussions with experts in the field.

Conclusion:

Out of the 43 features from clinical and metabolic test results, 6 potential features were identified. Classification of PCOS with the feature set transformed using standardisation is done using various machine learning techniques such as Naïve Bayes, Logistic Regression, K-Nearest neighbor (KNN), Random Forest Classifier (RFC), Support Vector Machine (SVM), XGBOOST and MLP in Python. Results revealed that the most suitable and accurate method for the PCOS prediction is MLP with an accuracy of almost 92.07%.

This automated system, that we aim to build in the future, can act as an assistive tool for doctors and save considerable time in examining patients and hence reducing delay in diagnosing the risk of PCOS.